

# 프로그래밍 언어 활용

문제해결 시나리오

악성코드분석 및 모의해킹 전문가 양성과정

4조 김 다 승

제출일 : 2019년 9월 03일

## 차 례

1.	프로그램 개요	----- 3
1)	프로그램 소개	----- 3
2)	프로그램 상세설명	----- 3
(1)	게임 시나리오	----- 3
(2)	게임 진행 방식	----- 4
(3)	기능	----- 5
2.	프로그램 구현	----- 7
1)	제어문(조건문, 반복문)	----- 7
2)	함수	----- 15
3)	배열, 포인터	----- 26
4)	구조체	----- 26
5)	파일 입출력	----- 26
3.	프로그램 시현	----- 27
1)	빌드	----- 27
2)	시현 순서	----- 27
4.	프로젝트 후 나의 생각	----- 29
5.	향후 계획	----- 29

## 1. 프로그램 개요

### 1) 프로그램 소개

- ▷ 제목 : ANYPANG(애니팡)
- ▷ 장르 : 퍼즐(제거) 게임

### 2) 프로그램 상세설명

#### (1) 게임 시나리오


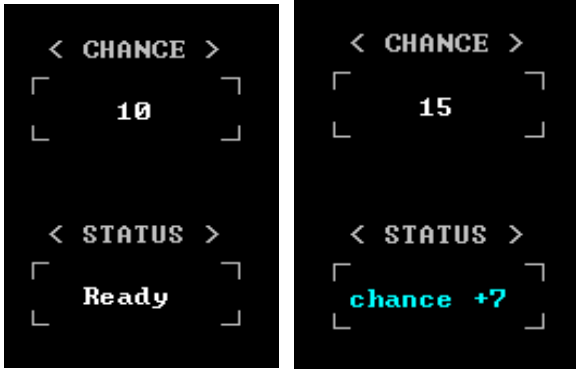
- ▷ 제목 : ANYPANG
- ▷ 장르 : 퍼즐 게임
- ▷ 컨트롤 : 마우스 클릭
- ▷ 기획한 계기 : 인터넷에서 검색하다가 테트리스나 뱀 게임 등 이미 나와있는 게임들을 제외한 나머지 게임을 생각하다 나온 게 퍼즐 게임이었고 팀원분께서 애니팡을 말씀해주셔서 시작하게 되었습니다.
- ▷ 애니팡과 다른 점
  - : 기존 애니팡을 모티브로 만들어진 게임이므로 퍼즐게임이기는 하나 애니팡은 손가락으로 퍼즐을 끌어서 같은 퍼즐끼리 3개 이상 만나게 되면 제거가 되지만 저희가 만든 게임은 끌기가 아닌 같은 색상으로 3개 이상 붙어있는 퍼즐을 클릭하면 제거가 되도록 만들었습니다.
- ▷ 맵 구성(게임 화면 구성)
  - ㉠ 맵 : 시작위치는 (10,2)로 정한 후,  
x(0~23), y(0~23)으로 0과 22인 경우 특수기호를 출력해서 맵의 벽을 구현했습니다.
  - ㉡ 퍼즐 : 맵의 벽을 제외한 나머지 공간에 난수 0~4(0,1,2,3,4)를 주어 생성하고 난수에 색상을 매크로로 정의하여 숫자가 출력되지 않고 색상을 가진 퍼즐이 출력되도록 했습니다.
  - ㉢ 점수판
    - ㉣ SCORE : 현재 점수
    - ㉤ BESTSCORE : 최고 점수
      - > 현재 점수가 최고 점수를 갱신하면 현재 점수와 최고 점수 둘 다 잠시동안 초록색으로 출력되도록 했습니다.
      - > 게임을 종료하고 다시 해도 최고 점수는 저장되어 0으로 초기화되지 않습니다.




- ㉔ 찬스 : 기본으로 주어지는 건 10번
  - ㉔ 3개 이상 10개 미만인 경우, 찬스 -1씩 감소 및 (잠시동안) 빨간색 출력
  - ㉔ 10번 안에 10개 이상 퍼즐 제거 시, (제거된 퍼즐 수 - 3)만큼의 찬스 증가 및 (잠시동안) 초록색 출력
- ㉔ 메뉴 : 게임 도중에 "[CLICK HERE] MENU"를 클릭 시,  
[CONTINUE], [RESET], [QUIT] 등 선택할 수 있는 창이 뜹니다.
  - ㉔ [CONTINUE] : 게임으로 돌아가기
  - ㉔ [RESET] : 새 게임 시작하기
  - ㉔ [QUIT] : 게임 종료하기

## (2) 게임 진행 방식

- ▷ 시작화면에서 ESC키를 제외한 나머지 키를 누르면 게임이 시작됩니다.
- ▷ 시작화면에서 ESC키를 누르면 게임 종료됩니다.
- ▷ 게임 시작 시, 마우스 클릭으로 게임으로 진행하면 됩니다.
- ▷ 찬스를 다 쓴 후, 게임 종료 시, 더 하고 싶다면 RETRY  
아니면 QUIT을 눌러 종료합니다.
- ▷ 최고 점수를 0으로 초기화하고 싶다면 score.txt 파일에  
최고 점수를 지우고 0으로 저장하면 됩니다.

### (3) 기능

기능	설명
찬스 제도	<div data-bbox="700 264 983 678" data-label="Image">  </div> <p>① 기본으로 주어지는 찬스(기회)는 총 10번입니다.          ② 마우스 클릭 한 번 당 -1씩 감소됩니다.          ③ 만약 퍼즐을 10개 이상 제거 시, (제거된 퍼즐 수 - 3)의 찬스가 증가됩니다.          ④ 감소할 때 빨간색으로 증가할 때 초록색으로 잠시 출력되도록 했습니다.          ⑤ 맵 밖에서 클릭 시 기회가 감소되지 않도록 했습니다.</p>
보너스	<div data-bbox="553 940 1131 1308" data-label="Image">  </div> <p>① 사용자의 상태 창으로 상태를 출력합니다.          ② 사용자가 마우스 클릭으로 퍼즐을 제거 시, 10개 이상 제거가 아니면 "chance -1"를 표시합니다.          ③ 만약 10개 이상 제거 시, "chance +(제거된 퍼즐 수 -3)"을 출력합니다.</p>

기능	설명
점수 측정	<div data-bbox="225 367 539 580" data-label="Image">  </div> <div data-bbox="569 264 782 600" data-label="Image">  </div> <p>① 현 점수(Score)가 최고 점수를 갱신한 경우, 현 점수와 최고 점수의 색상을 초록색으로 잠시 출력되도록 했습니다.</p> <p>② 최고 점수를 갱신한 게임 판이 종료되면 "☆☆BestScore☆☆"가 화면에 출력되도록 했습니다.</p> <p>③ 최고 점수를 넘긴 판이 끝나면 최고 점수에 현재 점수를 넘겨 파일에 저장하도록 하여 다음 판에서도 최고 점수가 초기화되지 않고 그대로 남도록 구성되었습니다.(파일의 점수를 0으로 변경하면 초기화 가능합니다)</p>
일 시 정 지	<div data-bbox="225 963 782 1382" data-label="Image">  </div> <p>① 게임 도중에 "[CLICK HERE] MENU"를 클릭 시, 메뉴화면에 [CONTINUE], [RESET], [QUIT] 선택화면이 출력됩니다.</p> <p>② Continue : 게임으로 돌아가기 Reset : 새 게임 시작하기 Quit : 게임 종료하기</p>
퍼 즐	<p>① 새 게임 시, 퍼즐 생성</p> <p>② 마우스 클릭 후, 같은 퍼즐인지 비교한 후 제거</p> <p>③ 제거된 퍼즐의 수를 인식</p> <p>④ 제거한 퍼즐만큼 위에서 아래로 새 퍼즐 생성하기</p>
마우스 인식	<p>① 마우스 클릭 시, 마우스 클릭한 좌표를 읽어 출력되지는 않지만 해당 좌표의 퍼즐이 마우스 클릭 후의 이벤트를 실행하도록 했습니다.</p>

## 2. 프로그램 구현

### 1) 제어문(조건문, 반복문)

제어문(조건문, 반복문)	함수 목록
1_Main.c의 while문	<p>① 게임 화면의 맵 구성에 필요한 함수 호출</p> <pre> while (1) {     PrintGameBoard(board);     MouseClickEvent(&amp;pos_x, &amp;pos_y);     row = pos_y;     col = pos_x / 2;     if (row == 21 &amp;&amp; col &gt;= -9 &amp;&amp; col &lt;= -2) {         MenuDrawer();     }     else if (row &gt;= 0 &amp;&amp; row &lt;= 20 &amp;&amp; col &gt;= 0 &amp;&amp; col &lt;= 20) {         target[row][col] = 1;         BoardEvent(board, temp_board, target);     }     ChanceDrawer();     WriteScore(); } </pre>
3_GameDrawer.c	<p>① IntroDrawer() : 시작화면 출력</p> <pre> Gotoxy(x + 0, y + 0); for (int i = 1; i &lt; 32; i++) {     if (i % 2 == 0) {         HIGH_GREEN printf("%s", intro1);     }     else if (i % 2 != 0) {         RED printf("%s", intro1);     } }  for (j = 4; j &gt;= 0; j--) {     B_x = 5, B_y = 9;     for (i = 4; i &gt;= j; i--) { // 아래에서 위로 출력+이동성         if (i &gt;= 1 &amp;&amp; i &lt; 4) { BLUE_GREEN }         else { GOLD }         Gotoxy(B_x, B_y); printf("%s", *(Any + i));         B_y--;         Sleep(20);     } }  for (j = 0; j &lt; 6; j++) {     B_x = 19, B_y = 5;     for (i = 0; i &lt; j; i++) { // 위에서 아래로 출력+ 이동성         if (i &gt;= 1 &amp;&amp; i &lt; 4) { GOLD }         else { BLUE_GREEN }         Gotoxy(B_x, B_y);         printf("%s", *(Pang + i));         B_y++; Sleep(20);     } }  Gotoxy(x + 0, y + 8); for (int i = 1; i &lt; 32; i++) {     if (i % 2 == 0) {         HIGH_GREEN printf("%s", intro2);     }     else if (i % 2 != 0) {         RED printf("%s", intro2);     } } </pre>

제어문(조건문, 반복문)	함수 목록
3_GameDrawer.c	<p>② PrintMap() : 맵의 벽을 출력</p> <pre> void PrintMap(int map[][SIZE_X]) {     for (int y = 0; y &lt; SIZE_Y; y++) {         for (int x = 0; x &lt; SIZE_X; x++) {             switch (map[y][x]) {                 case EMPTY: // 해당칸이 빈공간이면                     Gotoxy(x + POS_X_MAP, y + POS_Y_MAP);                     ORIGINAL printf(" ");                     break;                 case SKYLINE: // 해당칸이 천장이면                     Gotoxy(x + POS_X_MAP, y + POS_Y_MAP);                     ORIGINAL printf("▽");                     break;                 case WALL: // 해당칸이 벽이면                     Gotoxy(x + POS_X_MAP, y + POS_Y_MAP);                     ORIGINAL printf("▣");                     break;             }         }     } } </pre>
	<p>③ PrintGameBoard : 해당 좌표마다 블록 및 특수기호 출력</p> <pre> void PrintGameBoard(int board[][COL]) {     for (int y = 0; y &lt; ROW; y++) {         for (int x = 0; x &lt; COL; x++) {             switch (board[y][x]) {                 case EMPTY: // 해당칸이 삭제될때                     Gotoxy(x + POS_X_BLOCK, y + POS_Y_BLOCK);                     SKY_BLUE printf("◎"); ORIGINAL                     break;                 case RBLOCK: // 해당칸이 빨간 블록이면                     Gotoxy(x + POS_X_BLOCK, y + POS_Y_BLOCK);                     RED printf("■"); ORIGINAL                     break;                 case BBLOCK: // 해당칸이 파란 블록이면                     Gotoxy(x + POS_X_BLOCK, y + POS_Y_BLOCK);                     BLUE_GREEN printf("■"); ORIGINAL                     break;                 case GBLOCK: // 해당칸이 초록 블록이면                     Gotoxy(x + POS_X_BLOCK, y + POS_Y_BLOCK);                     HIGH_GREEN printf("■"); ORIGINAL                     break;                 case YBLOCK: // 해당칸이 노란 블록이면                     Gotoxy(x + POS_X_BLOCK, y + POS_Y_BLOCK);                     YELLOW printf("■"); ORIGINAL                     break;                 case ABLOCK: // 해당칸이 회색 블록이면                     Gotoxy(x + POS_X_BLOCK, y + POS_Y_BLOCK);                     GRAY printf("■"); ORIGINAL                     break;                 case SKYLINE: // 해당칸이 천장이면                     Gotoxy(x + POS_X_BLOCK, y + POS_Y_BLOCK);                     ORIGINAL printf("▽"); ORIGINAL                     break;                 case WALL: // 해당칸이 벽이면                     Gotoxy(x + POS_X_BLOCK, y + POS_Y_BLOCK);                     ORIGINAL printf("▣"); ORIGINAL                     break;             }         }     } } </pre>
	<p>④ ChanceDrawer() : 남은 기회를 출력</p> <pre> void ChanceDrawer() // 남은기회를 그림니다. {     Gotoxy(37, 4); WHITE printf("Xd ", chance); ORIGINAL     if (Com_chance &lt; chance) { // 전의 찬스 수 &lt; 지금 찬스 수         Gotoxy(37, 4); HIGH_GREEN printf("Xd ", chance); Sleep(150); ORIGINAL         Gotoxy(37, 4); WHITE printf("Xd ", chance); ORIGINAL     }     else if (Com_chance &gt;= chance) { // 전의 찬스 수 &gt; 지금 찬스 수         Gotoxy(37, 4); RED printf("Xd ", chance); Sleep(150); ORIGINAL         Gotoxy(37, 4); WHITE printf("Xd ", chance); ORIGINAL     } } </pre>
	<p>⑤ BonusDrawer() : 사용자 상태창 출력</p> <pre> void BonusDrawer() // 보너스획득 여부를 그림니다. {     if (bonus &gt; 0) {         Gotoxy(35, 10); SKY_BLUE printf("chance +Xd ", bonus); ORIGINAL     }     else if (bonus &lt; 0) {         Gotoxy(35, 10); GRAY printf ("chance Xd ", bonus); ORIGINAL     }     else {         Gotoxy(35, 10); WHITE printf(" Ready "); ORIGINAL     } } </pre>



제어문(조건문, 반복문)	함수 목록
3_GameDrawer.c	<p>⑥ HideDrawer() : 게임 중에 메뉴 클릭 시 검은화면 출력</p> <pre> void HideDrawer() {     for (int y = 0; y &lt; ROW; y++) { // 게임 영역 가리기         for (int x = 0; x &lt; COL; x++) {             Gotoxy(x + POS_X_BLOCK, y + POS_Y_BLOCK);             printf(" ");         }     } } </pre> <p>⑦ GameOverDrawer() : 게임 오버화면을 출력</p> <pre> void GameOverDrawer() { // 게임 오버화면을 그림     HideDrawer(); // 게임 내 블록 모두 지우기     if (bestscoreFlag == 1) {         Gotoxy(16, 9); printf(" ★★BestScore★★");         bestscoreFlag = 0;     }     Gotoxy(16, 12); printf(" [ ★GameOver★ ] ");     Gotoxy(16, 14); printf(" [RETRY] [QUIT] ");     EndManager();     Gotoxy(17, 9); printf(" ");     Gotoxy(16, 12); printf(" ");     Gotoxy(16, 14); printf(" "); } </pre> <p>⑧ WriteScore() : 파일에 있는 최고 점수를 저장</p> <pre> void WriteScore() {     if (yourscore &gt; bestscore) {         bestscoreFlag = 1;         bestscore = yourscore;         FILE* data = fopen("score.txt", "w");         if (!data) { // 파일에 에러가 뜨면             Gotoxy(0, 0);             printf("데이터 저장에 실패했습니다");         }         else {             fprintf(data, "%d", bestscore);             fclose(data);             //bestscoreFlag = 0;         }         Gotoxy(5, 10); HIGH_GREEN printf("%d ", bestscore); ORIGINAL         Gotoxy(5, 4); HIGH_GREEN printf("%d ", yourscore); Sleep(150); ORIGINAL         Gotoxy(5, 10); WHITE printf("%d ", bestscore); ORIGINAL         Gotoxy(5, 4); WHITE printf("%d ", yourscore); ORIGINAL     } } </pre>
4_BlockMaker.c	<p>① NewBlockMaker() : 새 게임 화면에서 퍼즐생성</p> <pre> void NewBlockMaker(int board[][COL]) {     for (int i = 0; i &lt; 21; i++) {         for (int j = 0; j &lt; 21; j++) {             board[i][j] = rand() % 5;         }     } } </pre> <p>② SetBlock() : 제거된 블록(퍼즐)의 수를 인식</p> <pre> void SetBlock(int board[][COL]) {     for (int col = 0; col &lt; 21; col++) {         for (int row = 0; row &lt; 21; row++) {             if (board[row][col] == -1) {                 for (int z = row; z &gt; 0; z--) {                     board[z][col] = board[z - 1][col];                 }                 board[0][col] = -1;             }         }     } } </pre> <p>③ ResetBlock() : 제거된 수만큼 새로운 블록(퍼즐) 생성</p> <pre> void ResetBlock(int board[][COL]) {     for (int i = 0; i &lt; 21; i++) {         for (int j = 0; j &lt; 21; j++) {             if (board[i][j] == -1)                 board[i][j] = rand() % 5;         }     } } </pre>

제어문(조건문, 반복문)	함수 목록
5_GameController.c	<p>① KeyboardControl() : 키보드 제어</p> <pre> void KeyboardControl() { // 게임 중 키보드 키 기능     int key;     if (_kbhit()) {         key = _getch();         if (key == 224) {             key = _getch();             switch (key) { // 방향키 동작x                 case LEFT:                     break;                 case RIGHT:                     break;                 case UP:                     break;                 case DOWN:                     break;             }         }         else {             switch (key) {                 case ESC: // ESC 누를 시                     system("cls");                     EndScreen();                     exit(0);             }         }     } } </pre> <p>② MouseClickEvent() : 마우스 클릭 인식</p> <pre> void MouseClickEvent(int* pos_x, int* pos_y) {     *pos_x = *pos_y = 0;     HANDLE hIn, hOut;     INPUT_RECORD rec;     DWORD dwNOER;      hIn = GetStdHandle(STD_INPUT_HANDLE);     hOut = GetStdHandle(STD_OUTPUT_HANDLE);     SetConsoleMode(hIn, ENABLE_PROCESSED_INPUT   ENABLE_MOUSE_INPUT);     while (1) {         ReadConsoleInput(hIn, &amp;rec, 1, &amp;dwNOER);         if (rec.EventType == MOUSE_EVENT) {             if (rec.Event.MouseEvent.dwButtonState &amp; FROM_LEFT_1ST_BUTTON_PRESSED) {                 *pos_x = rec.Event.MouseEvent.dwMousePosition.X - 22;                 *pos_y = rec.Event.MouseEvent.dwMousePosition.Y - 3;                 return;             }         }     } } </pre>
6_BlockDetector.c	<p>① InitTempboard() : 블록탐지용 임시배열 temp_board를 초기화</p> <pre> void InitTempboard() {     for (int i = 0; i &lt; ROW; i++) {         for (int j = 0; j &lt; COL; j++) {             temp_board[i][j] = 0;         }     } } </pre> <p>② InitTarget() : 블록탐지용 임시배열 target을 초기화</p> <pre> void InitTarget() {     for (int i = 0; i &lt; ROW; i++) {         for (int j = 0; j &lt; COL; j++) {             target[i][j] = 0;         }     } } </pre> <p>③ RemoveBlock() : 블록 제거</p> <pre> void RemoveBlock(int board[][COL], int temp_board[][COL]) {     for (int row = 0; row &lt; 21; row++) {         for (int col = 0; col &lt; 21; col++) {             if (temp_board[row][col] == 1) {                 board[row][col] = -1;             }         }     } } </pre>

제어문(조건문, 반복문)	함수 목록
6_BlockDetector.c	<p>④ TargetIndex() : 다음 타겟의 블록 위치</p> <pre> int TargetIndex(int temp_board[][COL], int target[][COL]) {     int check = 0;      for (int i = 0; i &lt; 21; i++)     {         for (int j = 0; j &lt; 21; j++) {             if (temp_board[i][j] == -1) {                 target[i][j] = 1;                 check = 1;             }         }     }      return check; } </pre>
	<p>⑤ TargetCount() : 블록 개수 세기</p> <pre> int TargetCount(int arr[][COL]) {     int count = 0;      for (int i = 0; i &lt; 21; i++) {         for (int j = 0; j &lt; 21; j++) {             if (arr[i][j] != 0) {                 count++;             }         }     }      return count; } </pre>
	<p>⑥ ArrayClear() : 배열 삭제</p> <pre> void ArrayClear(int arr[][COL]) {     for (int i = 0; i &lt; 21; i++) {         for (int j = 0; j &lt; 21; j++) {             arr[i][j] = 0;         }     } } </pre>
	<p>⑦ ArrayIndex() : 배열 위치</p> <pre> void ArrayIndex(int target[][COL], int* row, int* col) {     for (int i = 0; i &lt; 21; i++) {         for (int j = 0; j &lt; 21; j++) {             if (target[i][j] == 1) {                 *row = i;                 *col = j;             }         }     } } </pre>

## ⑧ CompareBlock() : 선택한 블록과 인접한 블록을 비교

```
void CompareBlock(int board[][COL], int temp_board[][COL], int target[][COL]) {
    int row, col, current_block;

    ArrayIndex(target, &row, &col);
    current_block = board[row][col];
    temp_board[row][col] = 1;

    for (int i = 0; i < 21; i++) {
        for (int j = 0; j < 21; j++) {
            if (target[i][j] != 0) {
                row = i;
                col = j;
            }
        }
    }

    if (row == 0)
    {
        if (col == 0)
        {
            if (current_block == board[row + 1][col] && temp_board[row + 1][col] == 0)
                temp_board[row + 1][col] = -1;
            if (current_block == board[row][col + 1] && temp_board[row][col + 1] == 0)
                temp_board[row][col + 1] = -1;
        }
        else if (col == 20)
        {
            if (current_block == board[row][col - 1] && temp_board[row][col - 1] == 0)
                temp_board[row][col - 1] = -1;
            if (current_block == board[row - 1][col] && temp_board[row - 1][col] == 0)
                temp_board[row - 1][col] = -1;
        }
        else
        {
            if (current_block == board[row][col - 1] && temp_board[row][col - 1] == 0)
                temp_board[row][col - 1] = -1;
            if (current_block == board[row][col + 1] && temp_board[row][col + 1] == 0)
                temp_board[row][col + 1] = -1;
            if (current_block == board[row + 1][col] && temp_board[row + 1][col] == 0)
                temp_board[row + 1][col] = -1;
        }
    }
}
```

```

else if (row == 20)
{
    if (col == 0)
    {
        if (current_block == board[row - 1][col] && temp_board[row - 1][col] == 0)
            temp_board[row - 1][col] = -1;
        if (current_block == board[row][col + 1] && temp_board[row][col + 1] == 0)
            temp_board[row][col + 1] = -1;
    }
    else if (col == 20)
    {
        if (current_block == board[row - 1][col] && temp_board[row - 1][col] == 0)
            temp_board[row - 1][col] = -1;
        if (current_block == board[row][col - 1] && temp_board[row][col - 1] == 0)
            temp_board[row][col - 1] = -1;
    }
    else
    {
        if (current_block == board[row][col - 1] && temp_board[row][col - 1] == 0)
            temp_board[row][col - 1] = -1;
        if (current_block == board[row][col + 1] && temp_board[row][col + 1] == 0)
            temp_board[row][col + 1] = -1;
        if (current_block == board[row - 1][col] && temp_board[row - 1][col] == 0)
            temp_board[row - 1][col] = -1;
    }
}
else if (col == 0)
{
    if (current_block == board[row - 1][col] && temp_board[row - 1][col] == 0)
        temp_board[row - 1][col] = -1;
    if (current_block == board[row + 1][col] && temp_board[row + 1][col] == 0)
        temp_board[row + 1][col] = -1;
    if (current_block == board[row][col + 1] && temp_board[row][col + 1] == 0)
        temp_board[row][col + 1] = -1;
}
else if (col == 20)
{
    if (current_block == board[row - 1][col] && temp_board[row - 1][col] == 0)
        temp_board[row - 1][col] = -1;
    if (current_block == board[row + 1][col] && temp_board[row + 1][col] == 0)
        temp_board[row + 1][col] = -1;
    if (current_block == board[row][col - 1] && temp_board[row][col - 1] == 0)
        temp_board[row][col - 1] = -1;
}
else
{
    if (current_block == board[row - 1][col] && temp_board[row - 1][col] == 0)
        temp_board[row - 1][col] = -1;
    if (current_block == board[row + 1][col] && temp_board[row + 1][col] == 0)
        temp_board[row + 1][col] = -1;
    if (current_block == board[row][col - 1] && temp_board[row][col - 1] == 0)
        temp_board[row][col - 1] = -1;
    if (current_block == board[row][col + 1] && temp_board[row][col + 1] == 0)
        temp_board[row][col + 1] = -1;
}
}

ArrayClear(target);
}

```

7\_GameManager.c

### ① ReadBestScore() : 최고점수 파일 읽어오기

```

void ReadBestScore() { // 최고점수 읽어오기
    FILE* data = fopen("score.txt", "r");
    if (data == 0) {
        bestscore = 0;
    }
    else {
        fscanf(data, "%d", &bestscore);
        fclose(data);
    }
}

```



제어문(조건문, 반복문)	함수 목록
7_GameManager.c	<p>② ResetBoard() : 게임판 새 시작하기</p> <pre> void ResetBoard() { // 맵+게임판 초기화     for (int y = 0; y &lt; SIZE_Y; y++) { // 빈 공간(블록이 채워질 공간)         for (int x = 0; x &lt; SIZE_X; x++)             map[y][x] = EMPTY;     }     for (int y = 0; y &lt; SIZE_Y; y++) {         for (int x = 0; x &lt; SIZE_X; x++) {             if (x == 0    x == SIZE_X - 1) // 좌-우 벽 그리기                 map[y][x] = WALL;             if (y == SIZE_Y - 1) // 하단 벽 그리기                 map[y][x] = WALL;             if (y == 0 &amp;&amp; x != 0 &amp;&amp; x != SIZE_X - 1) // 천장 그리기                 map[y][x] = SKYLINE;         }     } } </pre> <p>③ GameoverCheck() : 게임오버인지 확인 및 출력, 10개 이상 제거 시, 찬스증가</p> <pre> void GameoverCheck(int* chance) { // 게임오버 검사     Com_chance = *chance;     if (removed_block &gt;= 10) {         (*chance) += removed_block - 3;         bonus = removed_block - 3;         BonusDrawer();     }     else {         (*chance)--;         bonus = -1;         BonusDrawer();     }     if (!*chance) {         WriteScore();         ChanceDrawer();         GameOverDrawer(); // 게임 오버 화면을 그림         ResetGame(); // 게임초기화         return;     } } </pre> <p>④ PauseManager() : 게임 중 일시정지, 메뉴창 선택 시 다음진행</p> <pre> void PauseManager() {     while (1) {         MouseClickEvent(&amp;pos_x, &amp;pos_y);         row = pos_y;         col = pos_x / 2;         if (row == 11 &amp;&amp; col &gt;= 3 &amp;&amp; col &lt;= 8) {             break;         }         else if (row == 11 &amp;&amp; col &gt;= 10 &amp;&amp; col &lt;= 13) {             NewBlockMaker(board);             ResetGame();             break;         }         else if (row == 11 &amp;&amp; col &gt;= 15 &amp;&amp; col &lt;= 17) {             EndScreen();             break;         }     } } </pre> <p>⑤ EndManager() : 좌표 클릭 시, 게임 초기화 또는 게임 종료</p> <pre> void EndManager() {     while (1) {         MouseClickEvent(&amp;pos_x, &amp;pos_y);         row = pos_y;         col = pos_x / 2;         if (row == 11 &amp;&amp; col &gt;= 5 &amp;&amp; col &lt;= 8) {             NewBlockMaker(board);             ResetGame();             break;         }         else if (row == 11 &amp;&amp; col &gt;= 12 &amp;&amp; col &lt;= 14) {             EndScreen();             break;         }     } } </pre>

## 2) 함수(Call by Value, Call by Reference)

### (1) 1\_Main.c

```
#include "matching.h"

int main() {
    SetConsole();
    srand((unsigned)time(NULL)); rand();

    IntroDrawer();
    system("cls");
    ResetGame();
    NewBlockMaker(board);
    PrintMap(map);
    for (int i = 0; i < ROW; i++) {
        for (int j = 0; j < COL; j++)
            temp_board[i][j] = 0;
    }
    for (int i = 0; i < ROW; i++) {
        for (int j = 0; j < COL; j++)
            target[i][j] = 0;
    }
    row = col = pos_x = pos_y = 0;
    while (1) {
        PrintGameBoard(board);
        MouseClickEvent(&pos_x, &pos_y);
        row = pos_y;
        col = pos_x / 2;
        if (row == 21 && col >= -9 && col <= -2) {
            system("cls");
            EndScreen();
        }
        else if (row >= 0 && row <= 20 && col >= 0 && col <= 20) {
            target[row][col] = 1;
            BoardEvent(board, temp_board, target);
        }
        WriteScore();
        GameoverCheck(&chance);
        ChanceDrawer();
    }
}
```

① int main() : 메인 함수

## (2) 2\_ConsoleController.c

```
#include "matching.h"

void ConsoleSize() { // 콘솔창크기
    system("mode con cols=85 lines=30"); // 가로 85, 세로 30으로 설정
}

void RemoveScrollbar() { // 스크롤바 제거
    HANDLE handle = GetStdHandle(STD_OUTPUT_HANDLE);
    CONSOLE_SCREEN_BUFFER_INFO info;
    GetConsoleScreenBufferInfo(handle, &info);
    COORD new_size = {
        info.srWindow.Right - info.srWindow.Left + 1,
        info.srWindow.Bottom - info.srWindow.Top + 1
    };
    SetConsoleScreenBufferSize(handle, new_size);
}

void Gotoxy(int x, int y) { // 좌표이동
    COORD pos = { 2 + x, y }; // 가로는 두 칸씩(특수문자) 이동합니다.
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), pos);
}

void RemoveCursor() { // 커서제거
    CONSOLE_CURSOR_INFO curinfo = { 0, };
    curinfo.dwSize = 1;
    curinfo.bVisible = 0;
    SetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE), &curinfo);
}

void SetConsole() { //콘솔셋팅
    ConsoleSize();
    RemoveScrollbar();
    RemoveCursor();
}
```

- ① void ConsoleSize() : 콘솔 창 크기 제어
- ② void RemoveScrollbar() : 콘솔 창의 스크롤바 제거(출력X)
- ③ void Gotoxy(int x, int y) : 좌표 함수
- ④ void RemoveCursor() : 커서 깜빡이는 거 제거
- ⑤ void SetConsole() : 콘솔 관련 함수 호출해서 셋팅



### (3) 3\_GameDrawer.c

```
#include "matching.h"

void IntroDrawer() {
    int x = 5; int y = 3;
    int i = 0, j = 0, size = 0; int B_x = 0, B_y = 0;
    char* Tile = "■";
    int x1 = 0, x2 = 0;
    int x2 = 41, x2 = 41;
    int y1 = 0, y1 = 0;
    int y2 = 28, y2 = 28;

    char intro1[3] = "▼", intro2[3] = "▲";

    char* Any[5];
    Any[0] = "
■ ■ ■ ■ ■
";
    Any[1] = "
■ ■ ■ ■ ■
";
    Any[2] = "
■ ■ ■ ■ ■
";
    Any[3] = "
■ ■ ■ ■ ■
";
    Any[4] = "
■ ■ ■ ■ ■
";

    char* Pang[5];
    Pang[0] = "
■ ■ ■ ■ ■
";
    Pang[1] = "
■ ■ ■ ■ ■
";
    Pang[2] = "
■ ■ ■ ■ ■
";
    Pang[3] = "
■ ■ ■ ■ ■
";
    Pang[4] = "
■ ■ ■ ■ ■
";

    ORIGINAL
    for (x1, y1; x1 <= x2; x1++) {
        Gotoxy(x1, y1); printf("%s\n", Tile);
    } x1 = x2;
    for (x2, y1 + 1; y1 <= y2; y1++) {
        Gotoxy(x2, y1); printf("%s\n", Tile);
    } y1 = y2;
    for (x2 - 1, y2; x2 >= x1; x2--) {
        Gotoxy(x2, y2); printf("%s\n", Tile);
    }
    for (x1, y2 - 1; y2 >= y1 + 1; y2--) {
        Gotoxy(x1, y2); printf("%s\n", Tile);
    } y2--; x1++; y1++; x2--;

    Gotoxy(x + 0, y + 0);
    for (int i = 1; i < 32; i++) {
        if (i % 2 == 0) {
            HIGH_GREEN printf("%s", intro1);
        }
        else if (i % 2 != 0) {
            RED printf("%s", intro1);
        }
    }

    for (j = 4; j >= 0; j--) {
        B_x = 5, B_y = 9;
        for (i = 4; i >= j; i--) { // 아래에서 위로 출력+이동성
            if (i >= 1 && i < 4) { BLUE_GREEN }
            else { GOLD }
            Gotoxy(B_x, B_y); printf("%s", *(Any + i));
            B_y--;
            Sleep(20);
        }
    }

    for (j = 0; j < 6; j++) {
        B_x = 19, B_y = 5;
        for (i = 0; i < j; i++) { // 위에서 아래로 출력+이동성
            if (i >= 1 && i < 4) { GOLD }
            else { BLUE_GREEN }
            Gotoxy(B_x, B_y);
            printf("%s", *(Pang + i));
            B_y++; Sleep(20);
        }
    }

    Gotoxy(x + 0, y + 8);
    for (int i = 1; i < 32; i++) {
        if (i % 2 == 0) {
            HIGH_GREEN printf("%s", intro2);
        }
        else if (i % 2 != 0) {
            RED printf("%s", intro2);
        }
    }

    Gotoxy(x + 1, y + 17); BLUE_GREEN printf("===== HOW TO PLAY =====");
    Gotoxy(x + 1, y + 19); GOLD printf("      [MOUSE CLICK] POP BLOCKS");
    Gotoxy(x + 1, y + 21); GOLD printf("      [ESC] 메뉴 및 나가기");
    Gotoxy(x + 1, y + 23); BLUE_GREEN printf("=====");

    Gotoxy(x + 22, 24 + y); ORIGINAL printf("Made by 박종욱 차현표 김다솜"); ORIGINAL
    while (1) {
        if (_kbhit()) {
            if (_getch() == 27)
                EndScreen();
            else
                break;
        }
        ORIGINAL:
        Gotoxy(x + 9, y + 13); printf(">> PRESS ANY BUTTON << "); Sleep(500);
    }
}
```

```

        Gotoxy(x + 9, y + 13); printf("
"); Sleep(500);
    }
    while (!_kbhit())
        _getch(); // 버퍼 지우기
}

void PrintGameBoard(int board[][COL]) {
    for (int y = 0; y < ROW; y++) {
        for (int x = 0; x < COL; x++) {
            switch (board[y][x]) {
                case EMPTY: // 해당칸이 삭제될때
                    Gotoxy(x + POS_X_BLOCK, y + POS_Y_BLOCK);
                    SKY_BLUE printf("○"); ORIGINAL
                    break;
                case RBLOCK: // 해당칸이 빨간 블록이면
                    Gotoxy(x + POS_X_BLOCK, y + POS_Y_BLOCK);
                    RED printf("■"); ORIGINAL
                    break;
                case BBLOCK: // 해당칸이 파란 블록이면
                    Gotoxy(x + POS_X_BLOCK, y + POS_Y_BLOCK);
                    BLUE_GREEN printf("■"); ORIGINAL
                    break;
                case GBLOCK: // 해당칸이 초록 블록이면
                    Gotoxy(x + POS_X_BLOCK, y + POS_Y_BLOCK);
                    HIGH_GREEN printf("■"); ORIGINAL
                    break;
                case YBLOCK: // 해당칸이 노란 블록이면
                    Gotoxy(x + POS_X_BLOCK, y + POS_Y_BLOCK);
                    YELLOW printf("■"); ORIGINAL
                    break;
                case ABLOCK: // 해당칸이 회색 블록이면
                    Gotoxy(x + POS_X_BLOCK, y + POS_Y_BLOCK);
                    GRAY printf("■"); ORIGINAL
                    break;
                case SKYLINE: // 해당칸이 천장이면
                    Gotoxy(x + POS_X_BLOCK, y + POS_Y_BLOCK);
                    ORIGINAL printf("▽"); ORIGINAL
                    break;
                case WALL: // 해당칸이 벽이면
                    Gotoxy(x + POS_X_BLOCK, y + POS_Y_BLOCK);
                    ORIGINAL printf("▣"); ORIGINAL
                    break;
            }
        }
    }
}

```

```

void UIDrawer() {
    int x = 2, y = 2;

    ORIGINAL
    Gotoxy(1 + x, y); printf("[ SCORE ]"); // 현재점수 표시칸
    Gotoxy(x, 1 + y); printf("┌");
    Gotoxy(6 + x, 1 + y); printf("┌");
    Gotoxy(x, 3 + y); printf("┌");
    Gotoxy(6 + x, 3 + y); printf("┌");

    Gotoxy(33 + x, y); printf("[ CHANCE ]"); // 남은시간 표시칸
    Gotoxy(32 + x, 1 + y); printf("┌");
    Gotoxy(38 + x, 1 + y); printf("┌");
    Gotoxy(32 + x, 3 + y); printf("┌");
    Gotoxy(38 + x, 3 + y); printf("┌");

    Gotoxy(x, 6 + y); printf("[ BESTSCORE ]"); // 최고점수 표시칸
    Gotoxy(x, 7 + y); printf("┌");
    Gotoxy(6 + x, 7 + y); printf("┌");
    Gotoxy(x, 9 + y); printf("┌");
    Gotoxy(6 + x, 9 + y); printf("┌");

    GRAY
    Gotoxy(x + 26, 26 + y); printf("Made by 김다솜 박종욱 차형표");

    Gotoxy(x - 1, 18 + y); printf("- How TO PLAY - ");
    Gotoxy(x - 1, 20 + y); printf("[MOUSE CLICK] POP");
    Gotoxy(x - 1, 22 + y); printf("[CLICK HERE] EXIT");
    ORIGINAL
}

```

```

void ChanceDrawer() // 남은기회를 그립니다.
{
    static int Com_chance = 10; // 전의 찬스수
    Gotoxy(37, 4); ORIGINAL printf("%d ", chance);
    if (Com_chance < chance) { // 전의 찬스 수 < 지금 찬스 수
        Gotoxy(37, 4); HIGH_GREEN printf("%d ", chance); Sleep(100);
        Gotoxy(37, 4); ORIGINAL printf("%d ", chance);
    }
    else if (Com_chance > chance) { // 전의 찬스 수 > 지금 찬스 수
        Gotoxy(37, 4); RED printf("%d ", chance); Sleep(100);
        Gotoxy(37, 4); ORIGINAL printf("%d ", chance);
    }
    Com_chance = chance;
}

```

```

void HideDrawer() {
    for (int y = 0; y < ROW; y++) { // 게임 영역 가리기
        for (int x = 0; x < COL; x++) {
            Gotoxy(x + POS_X_BLOCK, y + POS_Y_BLOCK);
            printf(" ");
        }
    }
}

void GameOverDrawer() { // 게임 오버화면을 그림
    HideDrawer(); // 게임 내 블록 모두 지우기
    if (bestscoreFlag == 1) {
        Gotoxy(16, 9); printf(" ★★BestScore★★");
        bestscoreFlag = 0;
    }
    Gotoxy(16, 15); printf(" [ ★GameOver★ ] ");
    Gotoxy(16, 16); printf(" PRESS ANY BUTTON ");
    EndPauseManager(); // 버퍼 비우기
}

void WriteScore() { // 점수 저장
    if (yourscore > bestscore) {
        bestscoreFlag = 1;
        bestscore = yourscore;
        FILE* data = fopen("score.txt", "w");
        if (!data) { // 파일에 에러가 뜨면
            Gotoxy(0, 0);
            printf("데이터 저장에 실패했습니다");
        }
        else {
            fprintf(data, "%d", bestscore);
            fclose(data);
            //bestscoreFlag = 0;
        }
        Gotoxy(5, 4); HIGH_GREEN printf("%d ", yourscore); Sleep(100);
    }
    Gotoxy(5, 10); ORIGINAL printf("%d ", bestscore);
    Gotoxy(5, 4); ORIGINAL printf("%d ", yourscore);
}

```

- ① void IntroDrawer() : 게임 시작 화면 출력 및  
키보드 입력 시 다음 진행
- ② void EndScreen() : 게임 종료 시 화면
- ③ void PrintMap(int map[][SIZE\_X]) : 게임 맵 출력
- ④ void PrintGameBoard(int board[][COL]) : 블록 화면 출력
- ⑤ void UIDrawer() : 점수, 찬스, 보너스 표시칸 출력
- ⑥ void ChanceDrawer() : 찬스를 출력(감소 : red, 증가 : green)
- ⑦ void BonusDrawer() : 사용자 상태창 표시(찬스 증감 출력)
- ⑧ void HideDrawer() : 게임 중 메뉴 클릭 시 검은화면 출력
- ⑨ void MenuDrawer() : 메뉴출력(Continue, Reset, Quit)
- ⑩ void GameOverDrawer() : 게임오버화면 출력(Retry, Quit)  
최고점수 갱신 시 BestScore 출력
- ⑪ void WriteScore() : 현재 점수가 최고점수 갱신 시  
점수들을 초록색으로 잠시동안 출력 및  
해당 점수를 최고 점수로 파일에 저장

#### (4) 4\_BlockMaker.c

```
#include "matching.h"

void NewBlockMaker(int board[][COL]) {
    for (int i = 0; i < 21; i++) {
        for (int j = 0; j < 21; j++) {
            board[i][j] = rand() % 5;
        }
    }
}

void SetBlock(int board[][COL]) {
    for (int col = 0; col < 21; col++) {
        for (int row = 0; row < 21; row++) {
            if (board[row][col] == -1) {
                for (int z = row; z > 0; z--) {
                    board[z][col] = board[z - 1][col];
                }
                board[0][col] = -1;
            }
        }
    }
}

void ResetBlock(int board[][COL]) {
    for (int i = 0; i < 21; i++) {
        for (int j = 0; j < 21; j++) {
            if (board[i][j] == -1) {
                board[i][j] = rand() % 5;
            }
        }
    }
}
```

- ① void NewBlockMaker(int board[][COL])  
: 새 게임 시, 블록 생성
- ② void SetBlock(int board[][COL])  
: 마우스 클릭 후, 맵 안의 블록을 클릭했는지 인식
- ③ void ResetBlock(int board[][COL])  
: 블록을 제거한 후, 새 블록을 생성

## (5) 5\_GameController.c

```
#include "matching.h"

void KeyboardControl() { // 게임 중 키보드 키 기능(마우스로 충분할 경우 삭제. 현재 보류중)
    int key;
    if (_kbhit()) {
        key = _getch();
        if (key == 224) {
            key = _getch();
            switch (key) { // 방향키 동작x
                case LEFT:
                    break;
                case RIGHT:
                    break;
                case UP:
                    break;
                case DOWN:
                    break;
            }
        }
        else {
            switch (key) {
                case ESC: // ESC 누를 시
                    system("cls");
                    EndScreen();
                    exit(0);
            }
        }
    }
}

void MouseClickEvent(int* pos_x, int* pos_y) {
    *pos_x = *pos_y = 0;
    HANDLE hIn, hOut;
    INPUT_RECORD rec;
    DWORD dwNDR;

    hIn = GetStdHandle(STD_INPUT_HANDLE);
    hOut = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleMode(hIn, ENABLE_PROCESSED_INPUT | ENABLE_MOUSE_INPUT);

    while (1) {
        ReadConsoleInput(hIn, &rec, 1, &dwNDR);
        if (rec.EventType == MOUSE_EVENT) {
            if (rec.Event.MouseEvent.dwButtonState & FROM_LEFT_1ST_BUTTON_PRESSED) {
                *pos_x = rec.Event.MouseEvent.dwMousePosition.X - 22;
                *pos_y = rec.Event.MouseEvent.dwMousePosition.Y - 3;
                return;
            }
        }
    }
}
```

① void KeyboardControl() : 키보드 제어(인식) 함수

② void MouseClickEvent(int\* pos\_x, int\* pos\_y)

: 마우스 이벤트(인식) 함수



## (6) 6\_BlockDetector.c

```
#include "matching.h"

void BoardEvent(int board[][COL], int temp_board[][COL], int target[][COL]){
    int next;

    while (1) {
        CompareBlock(board, temp_board, target);
        next = TargetIndex(temp_board, target);

        if (next == 0) {
            if (TargetCount(temp_board) >= 3) {
                RemoveBlock(board, temp_board);
                removed_block = TargetCount(temp_board);
                yourscore += removed_block;
                PrintGameBoard(board);
                Sleep(100);
                SetBlock(board);
                ResetBlock(board);
            }
            ArrayClear(temp_board);
            break;
        }
    }
}

void CompareBlock(int board[][COL], int temp_board[][COL], int target[][COL]) { // 블록비교
    int row, col, current_block;

    ArrayIndex(target, &row, &col);
    current_block = board[row][col];
    temp_board[row][col] = 1;

    for (int i = 0; i < 21; i++) {
        for (int j = 0; j < 21; j++) {
            if (target[i][j] != 0) { // 0이 아니라면
                row = i;
                col = j;
            }
        }
    }

    if (row == 0)
    {
        if (col == 0)
        {
            if (current_block == board[row + 1][col] && temp_board[row + 1][col] == 0)
                temp_board[row + 1][col] = -1;
            if (current_block == board[row][col + 1] && temp_board[row][col + 1] == 0)
                temp_board[row][col + 1] = -1;
        }
        else if (col == 20)
        {
            if (current_block == board[row][col - 1] && temp_board[row][col - 1] == 0)
                temp_board[row][col - 1] = -1;
            if (current_block == board[row - 1][col] && temp_board[row - 1][col] == 0)
                temp_board[row - 1][col] = -1;
        }
        else
        {
            if (current_block == board[row][col - 1] && temp_board[row][col - 1] == 0)
                temp_board[row][col - 1] = -1;
            if (current_block == board[row][col + 1] && temp_board[row][col + 1] == 0)
                temp_board[row][col + 1] = -1;
            if (current_block == board[row + 1][col] && temp_board[row + 1][col] == 0)
                temp_board[row + 1][col] = -1;
        }
    }
    else if (row == 20)
    {
        if (col == 0)
        {
            if (current_block == board[row - 1][col] && temp_board[row - 1][col] == 0)
                temp_board[row - 1][col] = -1;
            if (current_block == board[row][col + 1] && temp_board[row][col + 1] == 0)
                temp_board[row][col + 1] = -1;
        }
        else if (col == 20)
        {
            if (current_block == board[row - 1][col] && temp_board[row - 1][col] == 0)
                temp_board[row - 1][col] = -1;
            if (current_block == board[row][col - 1] && temp_board[row][col - 1] == 0)
                temp_board[row][col - 1] = -1;
        }
        else
        {
            if (current_block == board[row][col - 1] && temp_board[row][col - 1] == 0)
                temp_board[row][col - 1] = -1;
            if (current_block == board[row][col + 1] && temp_board[row][col + 1] == 0)
                temp_board[row][col + 1] = -1;
            if (current_block == board[row - 1][col] && temp_board[row - 1][col] == 0)
                temp_board[row - 1][col] = -1;
        }
    }
    else if (col == 0)
    {
        if (current_block == board[row - 1][col] && temp_board[row - 1][col] == 0)
            temp_board[row - 1][col] = -1;
        if (current_block == board[row + 1][col] && temp_board[row + 1][col] == 0)
            temp_board[row + 1][col] = -1;
        if (current_block == board[row][col + 1] && temp_board[row][col + 1] == 0)
            temp_board[row][col + 1] = -1;
    }
    else if (col == 20)
    {
        if (current_block == board[row - 1][col] && temp_board[row - 1][col] == 0)
            temp_board[row - 1][col] = -1;
    }
}
```

```

        if (current_block == board[row - 1][col] && temp_board[row - 1][col] == 0)
            temp_board[row - 1][col] = -1;
        if (current_block == board[row + 1][col] && temp_board[row + 1][col] == 0)
            temp_board[row + 1][col] = -1;
        if (current_block == board[row][col - 1] && temp_board[row][col - 1] == 0)
            temp_board[row][col - 1] = -1;
    }
    else
    {
        if (current_block == board[row - 1][col] && temp_board[row - 1][col] == 0)
            temp_board[row - 1][col] = -1;
        if (current_block == board[row + 1][col] && temp_board[row + 1][col] == 0)
            temp_board[row + 1][col] = -1;
        if (current_block == board[row][col - 1] && temp_board[row][col - 1] == 0)
            temp_board[row][col - 1] = -1;
        if (current_block == board[row][col + 1] && temp_board[row][col + 1] == 0)
            temp_board[row][col + 1] = -1;
    }

    ArrayClear(target);
}

int TargetIndex(int temp_board[][COL], int target[][COL]) {
    int check = 0;

    for (int i = 0; i < 21; i++)
    {
        for (int j = 0; j < 21; j++) {
            if (temp_board[i][j] == -1) {
                target[i][j] = 1;
                check = 1;
            }
        }
    }

    return check;
}

int TargetCount(int arr[][COL]) {
    int count = 0;

    for (int i = 0; i < 21; i++) {
        for (int j = 0; j < 21; j++) {
            if (arr[i][j] != 0) {
                count++;
            }
        }
    }

    return count;
}

void RemoveBlock(int board[][COL], int temp_board[][COL]) { //나중에 게임점수매니저 구현할 때 연계해야함
    for (int row = 0; row < 21; row++) {
        for (int col = 0; col < 21; col++) {
            if (temp_board[row][col] == 1)
                board[row][col] = -1;
        }
    }
}

void ArrayClear(int arr[][COL]) {
    for (int i = 0; i < 21; i++) {
        for (int j = 0; j < 21; j++) {
            arr[i][j] = 0;
        }
    }
}

void ArrayIndex(int target[][COL], int* row, int* col) {
    for (int i = 0; i < 21; i++) {
        for (int j = 0; j < 21; j++) {
            if (target[i][j] == 1) {
                *row = i;
                *col = j;
            }
        }
    }
}

void EndPauseManager() { // 버퍼주는용도
    int garbagy;
    Sleep(500);
    while (_kbhit())
        _getch();
    garbagy = _getch();
} // 퍼즈/종료 버퍼비우기

```

- ① void InitTempboard()  
: 블록탐지용 임시배열 temp\_board 초기화
- ② void InitTarget()  
: 블록탐지용 임시배열 target 초기화
- ③ void BoardEvent(int board[][COL], int temp\_board[][COL],  
                  int target[][COL])  
: 게임(board) 관련 모든 이벤트 함수 호출
- ④ void CompareBlock(int board[][COL],  
                      int temp\_board[][COL], int target[][COL])  
: 클릭한 블록과 인접한 블록을 비교
- ⑤ void TargetIndex(int temp\_board[][COL], int target[][COL])  
: 선택한 블록의 다음 블록 위치
- ⑥ void TargetCount(int arr[][COL])  
: 블록의 개수 세기(같은 블록인 경우)
- ⑦ void RemoveBlock(int board[][COL], int temp\_board[][COL])  
: 블록 제거
- ⑧ void ArrayClear(int arr[][COL]) : 배열 삭제
- ⑨ void ArrayIndex(int target[][COL], int\* row, int\* col)  
: 배열의 위치



## (7) 7\_GameManager.c

```
#include "matching.h"

void GameoverCheck(int* chance) { // 게임오버를 검사합니다. 알고리즘 재정립 필요
    if (removed_block >= 8)
        (*chance) += removed_block - 3; // chance = 제거한 블록 수 -3
    else
        (*chance)--;

    if (!*chance) {
        GameOverDrawer(); // 게임 오버 화면을 그림
        ResetGame(); // 게임초기화
        return;
    }
}

void ReadBestScore() { // 최고점수 읽어오기
    FILE* data = fopen("score.txt", "r");
    if (data == 0) {
        bestscore = 0;
    }
    else {
        fscanf(data, "%d", &bestscore);
        fclose(data);
    }
}

void ResetGame() { // 게임 초기화
    ReadBestScore(); // 최고점수 불러옴
    ResetBoard(); // 맵+게임판 초기화
    UIDrawer(); // UI 출력
    chance = 10;
    yourscore = 0;
    removed_block = 0;
    ChanceDrawer();
    WriteScore(); // 게임 점수 표시
}

void ResetBoard() { // 맵+게임판 초기화
    for (int y = 0; y < SIZE_Y; y++) { // 빈 공간(블록이 채워질 공간)
        for (int x = 0; x < SIZE_X; x++)
            map[y][x] = EMPTY;
    }
    for (int y = 0; y < SIZE_Y; y++) {
        for (int x = 0; x < SIZE_X; x++) {
            if (x == 0 || x == SIZE_X - 1) // 좌-우 벽 그리기
                map[y][x] = WALL;
            if (y == SIZE_Y - 1) // 하단 벽 그리기
                map[y][x] = WALL;
            if (y == 0 && x != 0 && x != SIZE_X - 1) // 천장 그리기
                map[y][x] = SKYLINE;
        }
    }
}
```

- ① void GameoverCheck(int\* Chance)  
: 게임오버 확인 및 10개 이상 블록을 제거 시  
( 제거한 블록 수 - 3 )만큼 찬스 증가
- ② void ReadBestScore() : 파일에서 최고점수 읽어오기
- ③ void ResetGame() : 게임 초기화
- ④ void ResetBoard() : 게임 초기화 시, 게임판도 초기화
- ⑤ void PauseManager() : 일시정지 후, 메뉴에서  
해당 좌표를 클릭 시 다음 진행
- ⑥ void EndManager() : 해당 좌표 클릭 시,  
게임 초기화 또는 게임 종료

### 3) 배열, 포인터

배열, 포인터	목록
matching.h (전역변수)	① int map[SIZE_Y][SIZE_X] : 맵 ② int board[ROW][COL] : 게임보드 ③ int temp_board[ROW][COL] : 블록탐지에 사용될 임시배열 ④ int target[ROW][COL] : 블록탐지에 사용될 임시배열
3_GameDrawer.c (지역변수)	① char intro1[3] = "▼", intro2[3] = "▲"; : 게임 타이틀 ② char* Any[5]; char* Pang[5]; : 게임 타이틀

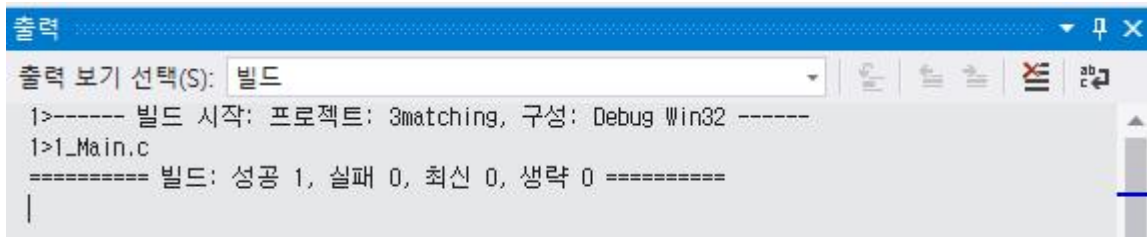
### 4) 구조체 : X

### 5) 파일 입출력

파일 입출력	목록
3_GameDrawer.c	① WriteScore() : 최고 점수 갱신 시, 현재 점수를 파일에 최고 점수로 저장  <pre> void WriteScore() {     if (yourscore &gt; bestscore) {         bestscoreFlag = 1;         bestscore = yourscore;         FILE* data = fopen("score.txt", "w");         if (!data) { // 파일에 에러가 뜨면             Gotaxy(0, 0);             printf("데이터 저장에 실패했습니다");         }         else {             fprintf(data, "%d", bestscore);             fclose(data);             //bestscoreFlag = 0;         }          Gotaxy(5, 10); HIGHGREEN printf("%d ", bestscore); ORIGINAL         Gotaxy(5, 4); HIGHGREEN printf("%d ", yourscore); Sleep(150); ORIGINAL     }     Gotaxy(5, 10); WHITE printf("%d ", bestscore); ORIGINAL     Gotaxy(5, 4); WHITE printf("%d ", yourscore); ORIGINAL } </pre>
7_GameManager.c	① ReadBestScore() : 파일에서 최고 점수를 읽어오기  <pre> void ReadBestScore() { // 최고점수 읽어오기     FILE* data = fopen("score.txt", "r");     if (data == 0) {         bestscore = 0;     }     else {         fscanf(data, "%d", &amp;bestscore);         fclose(data);     } } </pre>

### 3. 프로그램 시현

#### (1) 빌드



```
출력
출력 보기 선택(S): 빌드
1>----- 빌드 시작: 프로젝트: 3matching, 구성: Debug Win32 -----
1>1_Main.c
===== 빌드: 성공 1, 실패 0, 최신 0, 생략 0 =====
|
```

#### (2) 시현 순서

(1) Matching.h(헤더파일) : 전처리기, 필요한 헤더 파일 불러오기 및 변수, 함수 선언

(2) main() : 메인 함수 시작

(3) SetConsole() : 콘솔 창 제어하여 출력

- ▷ ConsoleSize() : 콘솔 창 크기 조절
- ▷ RemoveScrollbar() : 콘솔 창의 스크롤 바 제거
- ▷ RemoveCurser() : 화면에서 커서 표시X

(4) IntroDrawer() : 시작 화면 출력

- ▷ \_kbhit() & \_getch() : 키보드 인식
- ▷ Gotoxy() : 좌표 이동

(5) ResetGame() : 게임 화면 출력

- ▷ ReadBestScore() : 최고점수 불러옴
- ▷ ResetBoard() : 맵+게임판 초기화
- ▷ UIDrawer() : UI 출력
- ▷ chance = 10 : 찬스를 10으로 초기화
- ▷ yourscore = 0 : 현재 점수 0으로 초기화
- ▷ removed\_block = 0 : 제거한 블록 초기화
- ▷ bonus = 0 : 상태창 초기화
- ▷ ChanceDrawer() : 찬스 출력
- ▷ BonusDrawer() : 상태창 출력
- ▷ WriteScore() : 게임 점수 표시 및 최고 점수 저장

(6) NewBlockMaker() : 블록 생성(난수 0,1,2,3,4 준다)

(7) PrintMap() : 게임 맵 출력

(8) InitTempboard() : 임시배열 초기화

(9) InitTarget() : 임시배열 초기화

(10) PrintGameBoard() : 게임화면(게임판) 출력

(11) MouseClickEvent() : 마우스 이벤트(인식) 활성화

(12) MenuDrawer() : 게임 중 해당 좌표를 클릭하면 메뉴화면을 실행

▷ HideDrawer() : 메뉴 창이 출력되기 전에 검은바탕을 출력

▷ PauseManger() : 게임 일시정지 및 세 가지 선택 사항

㉠ Break : PauseManger()를 빠져나온다.

㉡ NewBlockMaker() : 새 블록을 생성

ResetGame() : 새 게임 시작

Break : PauseManger()를 빠져나온다.

㉢ EndScreen() : 게임 종료 화면

Break : PauseManger()를 빠져나온다.

(13) BoardEvent() : 사용자가 블록을 클릭한 다음, 좌표를 가져와서 실행

▷ CompareBlock() : 클릭한 블록과 인접한 블록들을 비교

▷ TargetIndex() : 같은 색상의 다음 블록이 있다면 다음 블록 위치를 알려줌

▷ TargetCount() : 같은 색상의 블록 수 세기

▷ RemoveBlock() : 같은 색상의 블록 수가 3개 이상이라면 블록 제거

▷ PrintGameBoard() : 다시 게임판 출력

▷ SetBlock() : 제거되는 블록을 인식

▷ ResetBlock() : 인식한 후, 제거된 블록 수만큼 채워주기

▷ GameoverCheck()

: 제거된 블록의 수가 10개 이상이라면 찬스를

(제거된 블록의 수 - 3)만큼 증가 아니라면 -1씩 감소하고 찬스 출력

만약 찬스가 없다면(0이라면) 점수를 출력 후, 찬스 출력, 게임오버 화면을 출력

게임오버 화면에서 "Retry"를 선택하면 게임오버화면을 빠져나와서 새 게임 시작

㉠ WriteScore() : 점수 출력 및 최고 점수 갱신 시 파일에 최고점수 저장

㉡ ChanceDrawer() : 찬스(기회) 출력

㉢ GameOverDrawer() : 게임오버 화면

㉣ HideDrawer() : 게임오버 화면을 출력하기 전 검은화면 출력

㉤ EndManager() : 해당 좌표를 클릭 시, 새 게임 시작 혹은 게임 종료

> NewBlockMaker() : 새 블록 생성

ResetGame() : 새로운 게임 시작

Break : EndManager()를 빠져나온다.

> EndScreen() : 게임 종료 화면 출력

Break : EndManager()를 빠져나온다.

© ArrayClear() : 배열 초기화 및 삭제

(14) Chance Drawer() : 찬스 출력

(15) WriteScore() : 점수 출력 및 최고점수를 갱신 시 파일에 점수 저장

## 5. 프로젝트 후 나의 생각

: 이번 프로젝트를 하면서 처음보는 함수들이 많아서 인터넷에 많이 검색하여 모방도 배움이라 생각하고 차츰 배워나갔습니다.

길다고 하면 길고 짧다고 하면 짧은 기간 동안 3명이 1팀으로 모여 프로젝트를 하는 것도 처음이었고 제가 게임을 만든다는 생각도 해보지 못해서 많이 생소했습니다.

게임을 많이 해보지 않아서 이런 게임이 있지 하면서도 실제로는 해보지 않아 어떤 기능이 있으며 어떤 점수제와 보너스, 아이템 등 관련 게임 지식이 부족해서 이번 프로젝트를 하는 데 팀원들에게 많은 도움이 되지 못해서 아쉬움만 생깁니다.

또한, C언어로 구현하는 능력에 있어서 미숙하여 대부분의 기능들을 다른 팀원들에게 부담하게 하여 제가 더 노력해야겠다는 생각을 하게 되었습니다.

다른 팀들의 프로젝트를 보면서 수업에서 배운 내용 가지고 충분히 구현할 수 있다는 것을 보고 이제까지 제가 코딩을 생각하면서 하지 않고 “외워서 했다” 라는 것을 많이 느끼게 되었습니다.

## 6. 향후 계획

: C언어 콘솔 창에서 정말 캐릭터를 구현할 수 없는지 아직 모르기 때문에 Google 등 검색엔진에 검색하면서 정보를 얻고 캐릭터를 구현하는 데 시도하려고 합니다. 만약 된다면 일정 점수가 되면 캐릭터의 게이지가 줄어들어서 쓰러지는 코드를 구현하려 합니다.

for문으로 출력하면서 아래에서 위로, 위에서 아래로 차츰 출력하는 거에 성공했지만 아직 오른쪽에서 왼쪽으로 출력하는 것은 아직 한 줄만 성공해서 틸틈이 제자리에서 오른쪽에서 왼쪽으로 이동하여 출력하는 것을 연습할 예정입니다.

위에 있는 과정을 포함하여 C언어를 좀 더 깊게 공부해서 생각한 것을 그대로 구현하되, 깔끔하게 코딩하는 작업을 연습해서 클린코드를 구현하는 것이 제 목표입니다.