보 고 서 #1

제 목: 4x4 퍼즐 게임 구현

과 목 명:	고급프로그래밍및실습
학 과:	스마트시스템소프트웨어학과
학 번:	20170404
이 름:	한종수
제 출 일:	2018년 3월 28일(수)
담당교수:	한 영 준

project1 - source_code.cpp*

source_code.cpp*

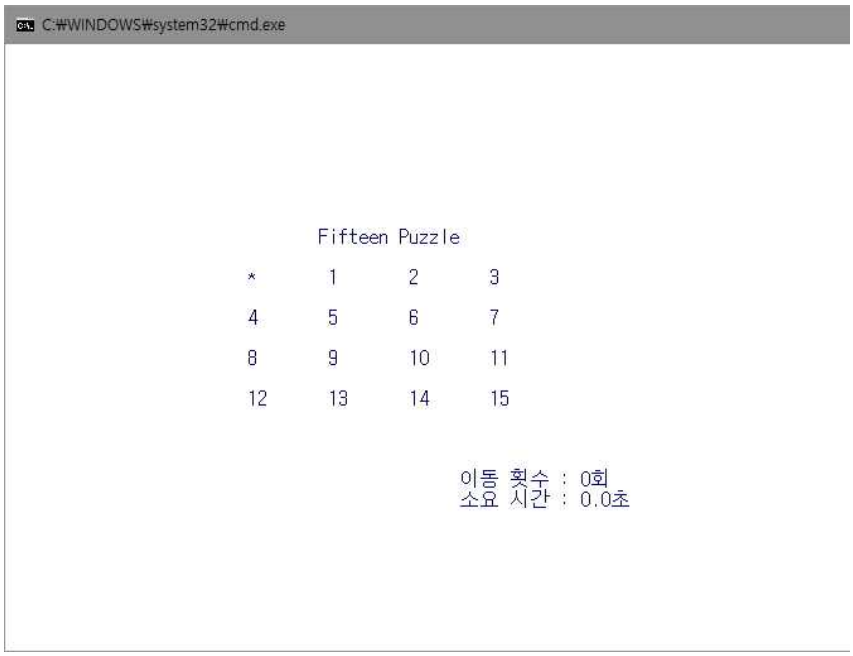
project1 | (전역 범위) | gameDraw(int(* gotmap)[DIM])

```cpp
#include <iostream>
#include <windows.h> //GetStdHandle(), Sleep(), SetCursorConsolePosition() 함수 사용
#include <conio.h> //_getch()와 kbhit() 함수 사용
#include <ctime> //clock() 함수 사용
#include <stdlib.h> //화면의 내용을 지우는 함수 system("cls") 사용

 //상수 선언부
#define ESC 27 //게임 종료
#define LEFT 75 //왼쪽 화살표 키 ASCII값
#define RIGHT 77 // 오른쪽 화살표 키 ASCII값
#define DOWN 80 //아래 화살표 키 ASCII값
#define UP 72 //위쪽 화살표 키 ASCII값
#define DIM 4 //4x4 퍼즐맵의 크기

//게임 Map 그리기 시작 위치: 필요하다면 출력 위치를 조정할 수 있음
#define MAP_STARTX 24
#define MAP_STARTY 9

//이동 회수 및 소요시간 출력 위치: 필요하다면 출력 위치를 조정할 수 있음
#define SCORE_STARTX 45
#define SCORE_STARTY 21

static int x, y; //빈칸의 위치
static int moveNum; //이동회수
static clock_t sTime; //시작 시간

//게임 시작 종료 안내 표시 위치지정
static int indentx = 11+MAP_STARTX;
static int indenty = 10+MAP_STARTY;

using namespace std;

 //함수 선언부
void gotoXY(short x, short y); //콘솔 화면에서 커서를 특정 위치로 이동
void init(int(*gotmap)[DIM]); //초기화 함수; 필요시에 매개변수를 추가함
void gameDraw(int (*gotmap)[DIM]); // 게임 맵과 반복회수, 시간 출력함수; MAP_STARTX, MAP_STARTY 상수를 참조해
                //항상 고정 위치(gotoXY함수 사용에 출력필요시에 매개변수를 추가함
                //                화면 출력 예시
                //               Fifteen Puzzle
                //                 1  2  9  3
                //                 6  8  7  4
                //                 10    5  15
                //                 13 14 12 11
                //
                //                 이동 회수: 10회
                //                 소요 시간: 15.2초

void puzzleMove(int action, int(*gotmap)[DIM]); //퍼즐 이동 함수; 필요시에 매개변수 추가함
void shuffle(int num, int(*gotmap)[DIM]);//퍼즐 맵 초기화; 50회 무작위 이동
int getAction(int (*gotmap)[DIM]);//모든 퍼즐이 제 위치에 있거나 ESC키가 눌러졌으면 정수 0을 반환
                                  // 방향키를 눌렀을 때는 해당 키값을 반환(LEFT, RIGHT, DOWN, UP)
int getDirectKey(); //방향키 획득 함수
void print_time_and_movenum(void); //게임이 시작된 후 시분초와 움직인 회수 표시하는 함수
int solvable(int(*gotmap)[DIM]); //섞여진 퍼즐이 풀수있는지 판단하는 함수

int main(void)
{
    int map[DIM][DIM]; //맵 크기를 지정
    int action = 1; //키보드의 값을 받는 변수

    init(map); //맨 처음 맵을 초기화 시킨다
    gameDraw(map); //맵을 그리는 함수를 호출한다.
    Sleep(1000); //초기상태의 맵을 보여주기 위해 1초간 멈춘다.
    gotoXY(indentx,indenty); //"섞는 중" 안내 표시를 위해 커서를 좌표로 이동한다.
    cout << "섞는 중";
    shuffle(50, map); //인자로 50을 넘겨 50번 섞이게끔 한다.

    gotoXY(indentx, indenty); //안내표시를 위해 이동한다.
    cout << "섞기 완료"; //안내 메시지 출력
    gotoXY(indentx,indenty); //동일
    cout << "게임 시작!"; //동일

    sTime = clock(); // 소요시간 계산을 위해 게임의 시작 시간을 기록해둔다.

    while (1) {
        action = getAction(map); // 퍼즐이 풀렸는지 판별하고 키를 입력받는다.
        if (action == 2) { //퍼즐을 맞추었을 때 그에 따른 안내를 하며 루프를 탈출한다.
            system("cls");
            gameDraw(map);
            gotoXY(indentx, indenty);
            cout << "퍼즐 완성!";
            break;
        }
        else if (action == 1) {
            gotoXY(MAP_STARTX, indenty); //잘못된 키를 입력받을때 그에 대한 안내를 표시한다.
            cout << "방향키 혹은 ESC키를 입력하세요!";
        }
        else if (action == 0) {//ESC키를 입력받았을때 안내하며 종료한다.
            system("cls");
            gameDraw(map);
            gotoXY(indentx, indenty);
            cout << "강제 종료!";
            break;
        }
        else {
            puzzleMove(action, map); //입력받은 방향으로 퍼즐을 이동시킨다.
            moveNum++; //퍼즐을 이동했기 때문에 이동 횟수를 1더한다.
        }
        gameDraw(map); //다시 맵을 갱신한다.
    }
```

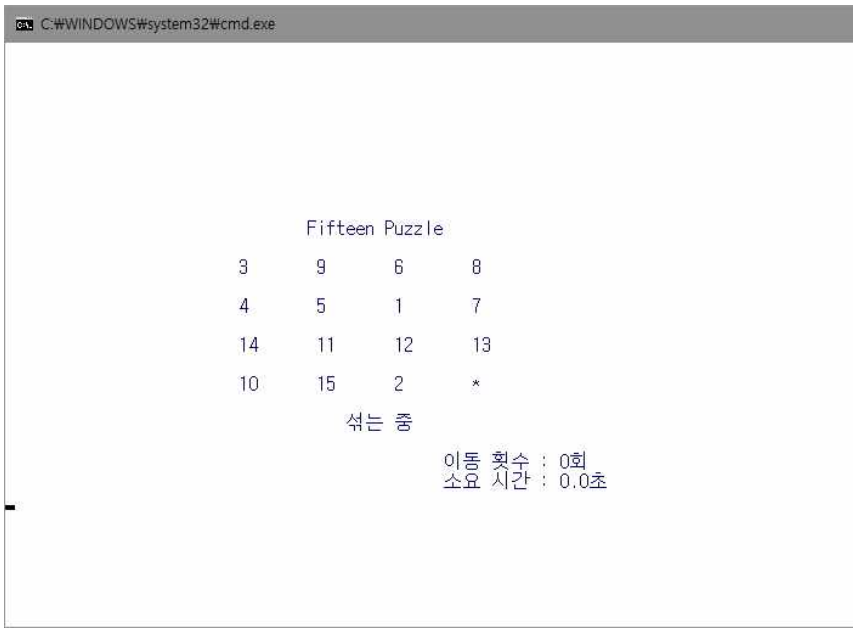
```
101
102         Sleep(1000); //화면을 지우기전에 잠깐의 대기시간을 준다.
103         system("cls"); //콘솔화면을 초기화시킨다.
104         init(map); //배열과 시간, 이동 횟수를 초기화 시킨다.
105         gameDraw(map); //맵을 갱신한다.
106         gotoXY(indentx, indenty); //게임종료 안내 메시지를 표시하기위해 이동한다.
107         cout << "게임 종료\n\n\n\n\n\n\n\n\n\n" ;
108
109         return 0;
110     }
111
112     //콘솔 화면에서 커서를 특정 위치로 이동
113     void gotoXY(short x, short y)
114     {
115         COORD Pos = { x, y };
116         SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), Pos);
117     }
118
119     // 그외 함수들을 정의함
120
121     void init(int(*gotmap)[DIM]) //초기화 함수; 필요시에 매개변수를 추가함
122     {
123         for (int i = 0; i < DIM*DIM; i++) // 맵의 값을 1,2,3,~의 순서로 초기화 시킨다.
124             *(gotmap[0] + i) = i;
125
126         sTime = clock(); //시작시간을 초기화 시킨다.
127         moveNum = 0; //이동 회수를 초기화시킨다.
128     }
129
130     void gameDraw(int(*gotmap)[DIM]) // 게임 맵과 반복회수, 시간 출력함수; MAP_STARTX, MAP_STARTY 상수를 참조해서
131     {                                //항상 고정 위치(gotoXY함수 사용에 출력필요시에 매개변수를 추가함
132         gotoXY(MAP_STARTX + 7, MAP_STARTY); //퍼즐의 이름을 출력하기 위해 이동한다.
133         cout << "Fifteen Puzzle";
134
135         for (int i = 0; i < DIM; i++) //배열에 저장된 값들을 출력한다.
136         {
137             gotoXY(MAP_STARTX, MAP_STARTY + 2 + i * 2); //간격을 2칸으로 출력한다.
138             for (int j = 0; j < DIM; j++)
139             {
140                 if (gotmap[i][j] == 0) //예외 처리를 통해 0의 자리의 값을 기호'*로 출력한다.
141                     cout << '*' << "\t";//0의 값을 '*'로 출력한다.
142                 else
143                     cout << gotmap[i][j] << "\t"; //일반 값들은 숫자를 출력한다.
144             }
145         }
146         print_time_and_movenum(); //소요된 시간과 이동횟수를 출력한다.
147     }
148
149     void puzzleMove(int action,int(*gotmap)[DIM]) //퍼즐 이동 함수; 필요시에 매개변수 추가함
150     {
151         int temp = 0; //값을 교환하기 위해 임시로 지정해 놓은 변수
152
153         switch (action) //키보드에서 입력받은 키보드 값을 바탕으로 배열의 값을 바꾼다.
154         {
155         case LEFT:
156             if (x >= 1) { //예외 처리를 통해 만약 '*'의 위치가 판을 벗어나지 않도록 한다.
157                 temp = gotmap[y][x];
158                 gotmap[y][x] = gotmap[y][x - 1];
159                 gotmap[y][x - 1] = temp;
160                 x--;
161             }
162             break;
163
164         case RIGHT:
165             if (x <= 2) {
166                 temp = gotmap[y][x];
167                 gotmap[y][x] = gotmap[y][x + 1];
168                 gotmap[y][x + 1] = temp;
169                 x++;
170             }
171             break;
172
173         case UP:
174             if (y >= 1) {
175                 temp = gotmap[y][x];
176                 gotmap[y][x] = gotmap[y-1][x];
177                 gotmap[y-1][x] = temp;
178                 y--;
179             }
180             break;
181
182         case DOWN:
183             if (y <= 2) {
184                 temp = gotmap[y][x];
185                 gotmap[y][x] = gotmap[y+1][x];
186                 gotmap[y+1][x] = temp;
187                 y++;
188             }
189             break;
190
191         default:
192             break; //아무것도 아니면 그냥 switch를 탈출한다.
193         }
194     }
195
196     void shuffle(int num, int(*gotmap)[DIM])
197     {
198         int numcheck[DIM*DIM]; //배열의 값이 중복되었는지 체크하기위한 맵과 동일한 크기의 배열을 생성한다.
199         int random = 0; //생성된 난수를 저장해 놓는 변수이다.
200
201         srand(time(NULL)); //난수의 시드 값을 바꾸어 줌으로써 난수 생성을 한다.
202
203         for (int k = 0; k < num; k++)
```

```
204     {
205         while (1) {
206             for (int i = 0; i < DIM*DIM; i++)
207                 numcheck[i] = 1; //난수 설정하기전에 중복체크 배열을 사용되기 전 상태인 '1'로 초기화 시킨다.
208 
209             for (int i = 0; i < DIM; i++)
210                 for (int j = 0; j < DIM; j++)
211                     while (1) {
212                         if ((numcheck[random = (rand() % (DIM*DIM))]) != 0) {
213                             if (random == 0) { //만약 난수의 값이 '*'의 좌표인 9이라면 그의 배열 좌표를 저장한다.
214                                 x = j; //'*'의 위치를 저장하기 위해 전역변수 x,y에 2차원 배열의 좌표를 저장한다.
215                                 y = i;
216                             }
217                             gotmap[i][j] = random;
218                             numcheck[random] = 0;
219                             break;
220                         } //생성된 난수가 사용되지 않은 난수이면 사용 표시를 하고 배열에 저장하며 루프를 탈출한다.
221                     }    // 난수가 사용 전이면 numcheck에 1의 형태로, 후면 0의 형태로 저장된다.
222 
223             if (solvable(gotmap) == 1) //퍼즐을 풀지 못하는 경우로 섞일 수 있다. 풀 수 있는지 판별하여
224                 break;                 //풀 수 있으면 1을 리턴하며 풀지못하면 0을 리턴하고 다시 섞는다.
225         }
226         sTime = clock(); //섞을 때는 시작 시간을 계속 초기화 시켜줌으로써 시간이 0으로 보이도록 한다.
227         gameDraw(gotmap); //맵을 갱신시킨다.
228         Sleep(100); //맵 변경이 보이도록 대기시간을 준다.
229     }
230 }
231 
232 int getAction(int(*gotmap)[DIM]) //모든 퍼즐이 제 위치에 있으면 2를 ESC키가 눌러졌으면 정수 0을 반환
233 {                                // 방향키를 눌렀을 때는 해당 키값을 반환(LEFT, RIGHT, DOWN, UP)
234     int i = 0; //맵의 값을 모두 비교하기 위해 임시변수를 설정해놓는다.
235 
236     while(1){
237         if (*(gotmap[0] + i) != i + 1) //맵의 값을 순서대로 늘어나는 값에 비교하여 만약 중간에 틀리면 루프를 탈출한다.
238             break;
239         else if(i == DIM*DIM - 2) //만약 순서대로 늘어나는 값이 배열의 크기의 마지막 값인 DIM*DIM-2 에 도달하면 2을 리턴한다.
240             return 2;
241         i++;
242     }
243 
244     switch (getDirectKey()) { //키를 입력받는 함수를 통해 키를 입력받고 키에 따라 값을 리턴한다.
245     case ESC:
246         return 0;
247     case LEFT:
248         return LEFT;
249     case RIGHT:
250         return RIGHT;
251     case DOWN:
252         return DOWN;
253     case UP:
254         return UP;
255     default:
256         return 1; // 방향키나 ESC키말고 다른키가 입력되면 1을 리턴한다.
257     }
258 }
259 
260 int getDirectKey(void)//방향키 획득 함수
261 {
262     int input = _getch(); //input 변수를 통해 입력된 키 값을 저장한다.
263 
264     if (input == 224) //방향키의 첫 1바이트는 224의 값을 가지며 이로 방향키인지를 판별한다.
265         return _getch(); //방향키는 2바이트 이므로 나머지 1바이트의 아스키 코드 값을 리턴한다.
266     else if (input == 27) //ESC의 아스키 코드 값을 입력받으면 ESC의 아스키 코드 값을 리턴한다.
267         return ESC;
268     else
269         return 1; // ESC와 방향키 이외의 키를 입력 받으면 1을 리턴한다.
270 }
271 
272 void print_time_and_movenum(void) //이동 횟수와 소요시간을 출력하는 함수
273 {
274     double time_flowed = (double)(clock() - sTime) / 1000; //현재 흐른 시간을 double형태로 소수점까지 저장한다.
275     int s = time_flowed, m = s / 60, h = m / 60; //시 분 초 단위로 시간을 나눈다.
276 
277     gotoXY(SCORE_STARTX, SCORE_STARTY); //이동 횟수와 소요 시간을 출력하는 좌표로 이동하여 정보를 출력한다.
278     cout << "이동 횟수 : " << moveNum << "회";
279     gotoXY(SCORE_STARTX, SCORE_STARTY + 1);
280     cout << "소요 시간 : ";
281 
282     if (h > 0) //1시간이 넘어가면 시 분 초 모두 출력한다.
283         cout << h << "시간 " << m << "분 ";
284     else if (m > 0)  //1분보다 많이 흐르고 1시간보다 적게 흐르면 분 초를 출력한다.
285         cout << m << "분 ";
286     else {}  //위의 경우에 해당하지 않으면 그냥 통과한다.
287 
288     //소수점 첫째 자리까지 소요시간 초를 출력한다.
289     cout.setf(ios::fixed);
290     cout.precision(1);
291     cout << (double)(s % 60 + (time_flowed - s)) << "초" << endl;
292     cout.unsetf(ios::fixed);
293 }
294 
295 int solvable(int(*gotmap)[DIM]) //무작위로 배열된 퍼즐이 풀수있는지 판별하는 함수
296 {
297     int inv_count = 0, k = 0,temp; //반전된 숫자 배열 세는 변수 등을 선언
298 
299     if (x != DIM - 1 || y != DIM - 1) //먼저 '*'표의 위치를 맨 아래 맨 오른쪽으로 이동시킨다.
300     {
301         temp = gotmap[DIM-1][DIM-1];
302         gotmap[DIM - 1][DIM - 1] = 0;
303         gotmap[y][x] = temp;
304         y = DIM - 1;
305         x = DIM - 1;
306     }
307 
308     for (int i = 0; i < DIM* DIM - 2; i++) //섞여진 수들을 일렬로 나열시키고 반전된 수의 수를 샌다.
309         for (int j = i + 1; j < DIM * DIM-1; j++)
310             if (*(gotmap[0] + i) > *(gotmap[0] + j))
311                 inv_count++;
312 
313     if (inv_count % 2 == 1) //만약 반전된 수의 수가 홀수 개이면 풀 수 없는 퍼즐이며 0을 리턴한다.
314         return 0;
315     else
316         return 1; //짝수 개이면 풀 수 있는 퍼즐이며 1을 리턴한다.
317 }
```

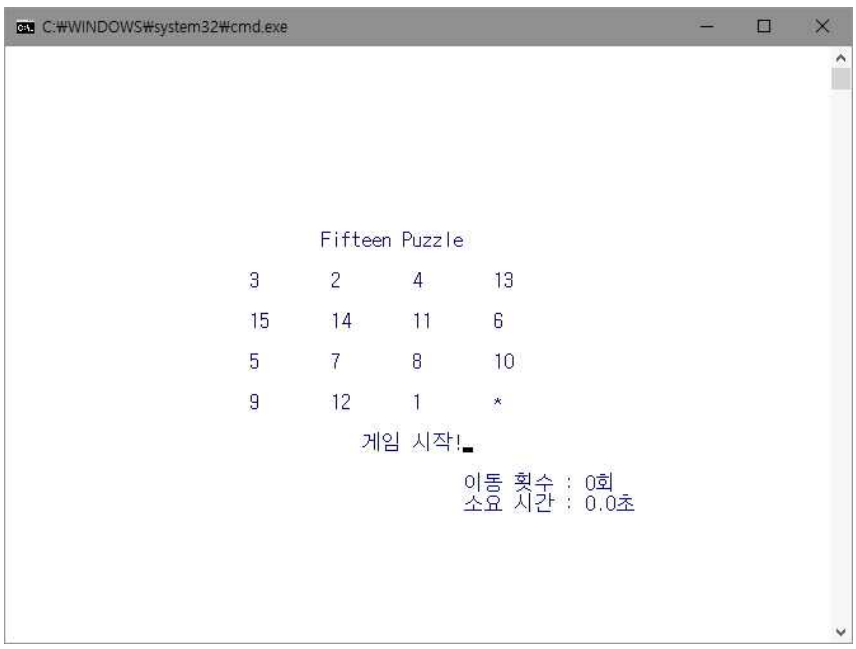
섞기 전 화면



섞는 화면



게임 시작 화면



게임 진행 화면

강제 종료 (ESC 키 사용)화면

퍼즐 완성 화면

게임 종료 화면