보 고 서 #2

제 목: 4x4 퍼즐 게임 구현

과 목 명:	고급프로그래밍및실습
학 과:	스마트시스템소프트웨어학과
학 번:	20170404
이 름:	한 종 수
제 출 일:	2018년 4월 16일(월)
담당교수:	한영준

1. 주석을 포함한 C++ 소스 코드를 프린트로 출력하여 제출

```
#include <iostream>
#include <windows.h> //GetStdHandle(), Sleep(), SetConsoleCursorPosition() 함수 사용
#include <conio.h> //getch()와 kbhit() 함수 사용
#include <ctime> //clock() 함수 사용

using namespace std;

#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 MDIM 4 //4x4 퍼즐맵의 크기
#define PDIM 7 //7x7 퍼즐의 크기

//게임 Map 그리기 시작 위치: 필요하다면 출력 위치를 조정할 수 있음
#define MAP_STARTX 15
#define MAP_STARTY 3

//이동 회수 및 소요시간 출력 위치: 필요하다면 출력 위치를 조정할 수 있음
#define SCORE_STARTX 34
#define SCORE_STARTY 36

static int numPat[16][7][7] =
{
    { 2,2,2,2,2,2,2,
    2,0,0,0,0,0,2,
    2,0,0,0,0,0,2,
    2,0,0,0,0,0,2, //빈칸
    2,0,0,0,0,0,2,
    2,0,0,0,0,0,2,
    2,2,2,2,2,2,2 },
    { 2,2,2,2,2,2,2,
    2,0,0,1,0,0,2,
    2,0,0,1,0,0,2,
    2,0,0,1,0,0,2,  //1
    2,0,0,1,0,0,2,
    2,0,0,1,0,0,2,
    2,2,2,2,2,2,2 },
    { 2,2,2,2,2,2,2,
    2,0,1,1,1,0,2,
    2,0,0,0,1,0,2,  //2
    2,0,1,1,1,0,2,
    2,0,1,0,0,0,2,
    2,0,1,1,1,0,2,
    2,2,2,2,2,2,2 },
    { 2,2,2,2,2,2,2,
    2,0,1,1,1,0,2,
    2,0,0,0,1,0,2,
    2,0,1,1,1,0,2,  //3
    2,0,0,0,1,0,2,
    2,0,1,1,1,0,2,
    2,2,2,2,2,2,2 },
    { 2,2,2,2,2,2,2,
    2,0,1,0,1,0,2,
    2,0,1,0,1,0,2,
    2,0,1,1,1,0,2,  //4
    2,0,0,0,1,0,2,
    2,0,0,0,1,0,2,
    2,2,2,2,2,2,2 },
    { 2,2,2,2,2,2,2,
    2,0,1,1,1,0,2,
    2,0,1,0,0,0,2,
    2,0,1,1,1,0,2,  //5
    2,0,0,0,1,0,2,
    2,0,1,1,1,0,2,
    2,2,2,2,2,2,2 },
    { 2,2,2,2,2,2,2,
    2,0,1,0,0,0,2,
    2,0,1,0,0,0,2,
    2,0,1,1,1,0,2,  //6
    2,0,1,0,1,0,2,
    2,0,1,1,1,0,2,
    2,2,2,2,2,2,2 },
    { 2,2,2,2,2,2,2,
    2,0,1,1,1,0,2,
    2,0,0,0,1,0,2,
    2,0,0,0,1,0,2,  //7
    2,0,0,0,1,0,2,
    2,0,0,0,1,0,2,
    2,2,2,2,2,2,2 },
    { 2,2,2,2,2,2,2,
    2,0,1,1,1,0,2,
    2,0,1,0,1,0,2,
    2,0,1,1,1,0,2,  //8
    2,0,1,0,1,0,2,
    2,0,1,1,1,0,2,
    2,2,2,2,2,2,2 },
    { 2,2,2,2,2,2,2,
    2,0,1,1,1,0,2,
    2,0,1,0,1,0,2,
    2,0,1,1,1,0,2,  //9
    2,0,0,0,1,0,2,
    2,0,0,0,1,0,2,
    2,2,2,2,2,2,2 },
    { 2,2,2,2,2,2,2,
    2,1,0,1,1,1,2,
    2,1,0,1,0,1,2,
    2,1,0,1,0,1,2,  //10
    2,1,0,1,0,1,2,
    2,1,0,1,1,1,2,
    2,2,2,2,2,2,2 },
    { 2,2,2,2,2,2,2,
```

Puzzle_Game - Puzzle_game.cpp*

Puzzle_game.cpp*

Puzzle_Game | puzzleGame | getCommand(void)

```
    2,1,0,1,0,1,2,
    2,1,0,1,1,1,2,
    2,2,2,2,2,2,2 },
    { 2,2,2,2,2,2,2,
    2,1,0,0,1,0,2,
    2,1,0,0,1,0,2,
    2,1,0,0,1,0,2,  //11
    2,1,0,0,1,0,2,
    2,1,0,0,1,0,2,
    2,2,2,2,2,2,2 },
    { 2,2,2,2,2,2,2,
    2,1,0,1,1,1,2,
    2,1,0,0,0,1,2,
    2,1,0,1,1,1,2,  //12
    2,1,0,1,0,0,2,
    2,1,0,1,1,1,2,
    2,2,2,2,2,2,2 },
    { 2,2,2,2,2,2,2,
    2,1,0,1,1,1,2,
    2,1,0,0,0,1,2,
    2,1,0,1,1,1,2,  //13
    2,1,0,0,0,1,2,
    2,1,0,1,1,1,2,
    2,2,2,2,2,2,2 },
    { 2,2,2,2,2,2,2,
    2,1,0,1,0,1,2,
    2,1,0,1,0,1,2,
    2,1,0,1,1,1,2,  //14
    2,1,0,0,0,1,2,
    2,1,0,0,0,1,2,
    2,2,2,2,2,2,2 },
    { 2,2,2,2,2,2,2,
    2,1,0,1,1,1,2,
    2,1,0,1,0,0,2,
    2,1,0,1,1,1,2,  //15
    2,1,0,0,0,1,2,
    2,1,0,1,1,1,2,
    2,2,2,2,2,2,2 }
};

enum COMMAND { Stop = 0, Left, Right, Down, Up, Non }; //방향키와 정지, 디폴트값 디코딩

class puzzleGame
{
public:

    puzzleGame(void); //초기화 함수; 필요시에 매개변수를 추가함
    void gameDraw(void); // 퍼즐로 구성되는 게임 맵과 반복회수, 시간 출력함수; MAP_STARTX, MAP_STARTY 상수를 참조
                         // 항상 고정 위치(gotoXY함수 사용)에 출력
                         // 필요시에 매개변수를 추가
    void puzzleDraw(int puzzle[][PDIM][PDIM], int x, int y, int k);// 하나의 퍼즐을 그리는 멤버함수
    void puzzleMove(int moveKey); //퍼즐 이동시키는 멤버 함수; 필요시에 매개변수 추가함
    void shuffle(int num);//퍼즐 맵 초기화; 50회 무작위 이동, 필요시에 인자 추가 할 수 있음
    void getCommand(void);//키값에 따라 명령어를 디코딩하는 멤버 함수
    int getAction(void) { return action; }; //private 멤버 변수 action을 class 밖에서 얻기 위한 함수
    void setsTime(double time) { sTime = time; }; //메인 함수에서 받은 시작 시간으로 sTime을 초기화하는 인라인 함수
    int solvable(void); //풀 수 있는 퍼즐인지 판별하는 함수

private:

    int map[MDIM][MDIM]; //퍼즐 배열
    int action; //어떤 행동을 수행할지 저장. (Stop, Left, Right, Down, Up, Non)
    int x, y; //빈칸의 위치
    int moveNum; //이동회수
    clock_t sTime; //시작 시간
    clock_t cTime; //현재 시간
};

void gotoXY(int x, int y); //콘솔 화면에서 커서를 특정 위치로 이동
int getDirectKey();

int main(void)
{
    puzzleGame gameBot; //class객체 생성
    int action = 0; //동작 저장
    gameBot.shuffle(50); //원하는 횟수만큼 퍼즐을 섞는다.
    gameBot.setsTime(clock()); // 소요시간 계산을 위해 게임의 시작 시간을 기록해둔다.
    while (1)
    {
        gameBot.getCommand(); //퍼즐을 검사하거나 키보드 입력을 통해어떤 행동을 수행할지 action에 저장한다.
        action = gameBot.getAction(); //class private의 action값을 불러온다.

        if (action == Stop)  //퍼즐을 맞추거나 ESC키를 입력받은 경우 종료한다.
            break;
        else
            gameBot.puzzleMove(action); //입력받은 방향으로 퍼즐을 이동시킨다.

        gameBot.gameDraw(); //다시 맵을 갱신한다.
    }

    system("cls"); //화면을 한번 지운다.
    gameBot.setsTime(clock()); //시간을 초기화한다.
    gameBot.gameDraw();
    gotoXY(SCORE_STARTX + 20, SCORE_STARTY);
    cout << "The game is over!!" << endl<<endl<<endl; //안내문을 출력한다.

    return 0;
}

puzzleGame::puzzleGame()
{
    for (int i = 0; i < MDIM*MDIM; i++)
        *(map[0] + i) = i+1;
    map[MDIM-1][MDIM-1] = 0;
```

Puzzle_Game - Puzzle_game.cpp*

Puzzle_game.cpp*

Puzzle_Game | puzzleGame | getCommand(void)

```
201  {
202      for (int i = 0; i < MDIM*MDIM; i++)
203          *(map[0] + i) = i+1;
204      map[MDIM-1][MDIM-1] = 0;
205      x = MDIM-1; y = MDIM-1;
206      // 맵의 값을 1,2,3,~ ,MDIM*MDIM-1,0의 순서로 초기화 시키고 빈칸의 위치를 마지막으로 초기화시킨다.
207      sTime = 0; //시작시간을 초기화 시킨다.
208      moveNum = 0; //이동 회수를 초기화시킨다.
209
210      //게임 초기화 코드 작성
211
212  }
213
214  void puzzleGame::puzzleDraw(int puzzle[][PDIM][PDIM], int x, int y, int k)
215  { //7*7의 숫자 형태와, 입력할 위치, 입력할 숫자를 입력받는다.
216      for (int i = 0; i< PDIM; i++)
217      {
218          gotoXY(x, y + i);
219          for (int j = 0; j< PDIM; j++)
220          {
221              if (puzzle[k][i][j] == 0)
222                  cout << "  ";
223              else if (puzzle[k][i][j] == 1)
224                  cout << "■";
225              else
226                  cout << "□";
227          }
228      } //숫자 형태의 모양에 따라 기호를 다르게 출력한다.
229  }
230
231  void puzzleGame::gameDraw(void)
232  {
233      gotoXY(SCORE_STARTX, SCORE_STARTY);
234      cout << "  이동 회수: " << moveNum << endl;
235      gotoXY(SCORE_STARTX, SCORE_STARTY + 1);
236      cout.setf(ios::fixed);
237      cout.precision(1);
238      cTime = clock();
239      cout << "  소요 시간: " << (cTime - sTime) / 1000.0 << "초" << endl;
240      cout.unsetf(ios::fixed);
241      //이동 회수 출력, 소요시간 소수점 첫째 자리까지 출력
242
243      gotoXY(MAP_STARTX + 20, MAP_STARTY);
244      cout << "   Fifteen Puzzle" << endl; //제목 출력
245
246      for (int i = 0; i< MDIM; i++)
247          for (int j = 0; j< MDIM; j++)
248              puzzleDraw(numPat, MAP_STARTX + (j * 15), MAP_STARTY + (i * 8) + 1, map[i][j]);
249                  //puzzleDraw 함수 호출해 퍼즐 출력.
250  }
251
252  void puzzleGame::puzzleMove(int moveKey)
253  {
254      int temp = 0;
255
256      switch (action) //getcommand를 통해 입력받은 키보드 값으로 퍼즐을 움직인다.
257      {
258      case Left:
259          if (x >= 1) { //예외 처리를 통해 빈칸의 위치가 판을 벗어나지 않도록 한다.
260              temp = map[y][x];
261              map[y][x] = map[y][x - 1];
262              map[y][x - 1] = temp;
263              x--; moveNum++; //퍼즐을 이동했기 때문에 이동 횟수를 1더한다.
264          }
265          break;
266
267      case Right:
268          if (x <= 2) {
269              temp = map[y][x];
270              map[y][x] = map[y][x + 1];
271              map[y][x + 1] = temp;
272              x++; moveNum++; //퍼즐을 이동했기 때문에 이동 횟수를 1더한다.
273          }
274          break;
275
276      case Up:
277          if (y >= 1) {
278              temp = map[y][x];
279              map[y][x] = map[y - 1][x];
280              map[y - 1][x] = temp;
281              y--; moveNum++; //퍼즐을 이동했기 때문에 이동 횟수를 1더한다.
282          }
283          break;
284
285      case Down:
286          if (y <= 2) {
287              temp = map[y][x];
288              map[y][x] = map[y + 1][x];
289              map[y + 1][x] = temp;
290              y++; moveNum++; //퍼즐을 이동했기 때문에 이동 횟수를 1더한다.
291          }
292          break;
293
294      default:
295          break; //아무것도 아니면 그냥 switch를 탈출한다.
296      }
297  }
298
299  void puzzleGame::getCommand(void)
300  {
301      int i = 0; //맵의 값을 모두 비교하기 위해 임시변수를 설정해놓는다.
302
```

Puzzle_Game - Puzzle_game.cpp*

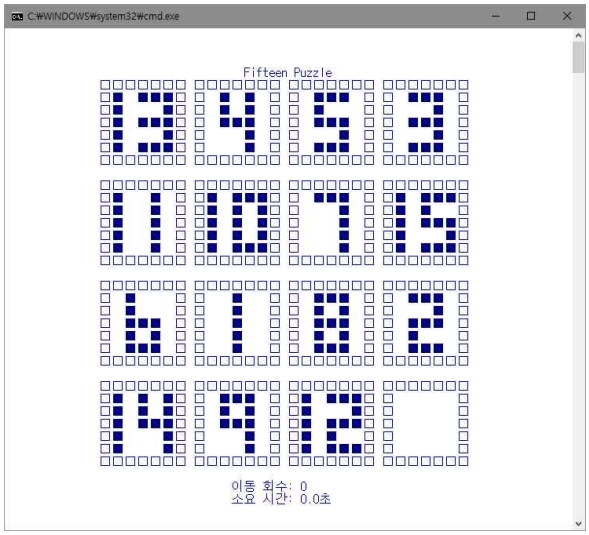
Puzzle_game.cpp*

Puzzle_Game | puzzleGame | getCommand(void)

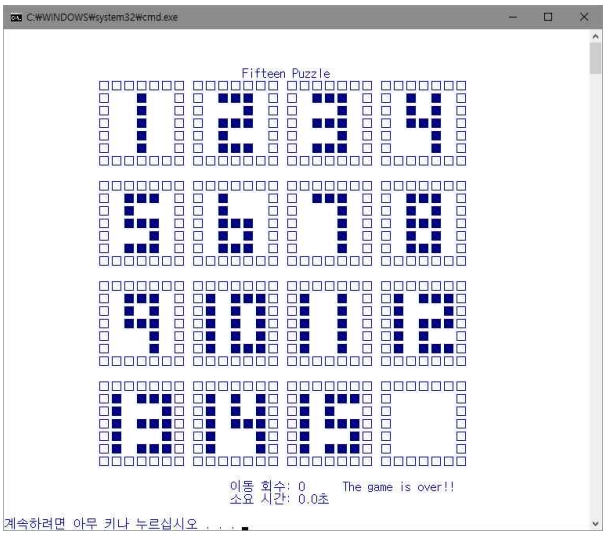
```
301     int i = 0; //맵의 값을 모두 비교하기 위해 임시변수를 설정해놓는다.
302
303     while (1) {
304         if (*(map[0] + i) != i + 1) //맵의 값을 순서대로 늘어나는 값에 비교하여 만약 중간에 틀리면 루프를 탈출한다.
305             break;
306         else if (i == MDIM*MDIM - 2) { //값이 배열의 크기의 마지막 값인 DIM*DIM-2 에 도달하면 명령을 수행하고 탈출
307             action = Stop; //정지 행동을 수행하게 끔 action을 Stop으로 초기화시킨다.
308             moveNum = 0; //정지 행동을 수행하기때문에 이동 회수를 초기화 시킨다.
309             break;
310         }
311         i++;
312     }
313     if (action != Stop) //퍼즐이 맞추어지지 않은 경우 키보드로 키를 입력받는다.
314         switch (getDirectKey()) { //getDirectKey 함수를 통해 키를 입력받고 명령어를 디코딩해 action에 저장한다.
315         case ESC:
316             action = Stop;
317             moveNum = 0;
318             break;
319         case LEFT:
320             action = Left;
321             break;
322         case RIGHT:
323             action = Right;
324             break;
325         case DOWN:
326             action = Down;
327             break;
328         case UP:
329             action = Up;
330             break;
331         default:
332             action = Non;
333             break; // 방향키나 ESC키말고 다른키가 입력되면 NON을 리턴한다.
334         }
335 }
336
337 void puzzleGame::shuffle(int num)
338 {
339     int numcheck[MDIM*MDIM]; //배열의 값이 중복되었는지 체크하기위한 맵과 동일한 크기의 배열을 생성한다.
340     int random = 0; //생성된 난수를 저장해 놓는 변수이다.
341
342     srand(time(NULL)); //난수의 시드 값을 바꾸어 줌으로써 난수 생성을 한다.
343
344     for (int k = 0; k < num; k++)
345     {
346         while (1) {
347             for (int i = 0; i < MDIM*MDIM; i++)
348                 numcheck[i] = 1; //난수 설정하기전에 중복체크 배열을 사용되기 전 상태인 '1'로 초기화 시킨다.
349
350             for (int i = 0; i < MDIM; i++)
351                 for (int j = 0; j < MDIM; j++)
352                     while (1) {
353                         if ((numcheck[random = (rand() % (MDIM*MDIM))]) != 0) {
354                             if (random == 0) { //만약 난수의 값이 빈칸의 좌표인 0이라면 그의 배열 좌표를 저장한다.
355                                 x = j; //빈칸의 위치를 저장하기 위해 멤버 변수 x,y에 2차원 배열의 좌표를 저장한다.
356                                 y = i;
357                             }
358                             map[i][j] = random;
359                             numcheck[random] = 0;
360                             break;
361                         } //생성된 난수가 사용되지 않은 난수이면 사용 표시를 하고 배열에 저장하며 루프를 탈출한다.
362                     }     // 난수가 사용 전이면 numcheck에 1의 형태로, 후면 0의 형태로 저장된다.
363
364             if (solvable() == 1)//퍼즐을 풀지 못하는 경우로 섞일 수 있다. 풀 수 있는지 판별하여
365                 break;                //풀 수 있으면 1을 리턴하며 풀지못하면 0을 리턴하고 다시 섞는다.
366         }
367         sTime = clock(); //섞을 때는 시작 시간을 계속 초기화 시켜줌으로써 시간이 0으로 보이도록 한다.
368         gameDraw(); //맵을 갱신시킨다.
369         Sleep(100); //맵 변경이 보이도록 대기시간을 준다.
370     }
371 }
372
373 int puzzleGame::solvable(void) //무작위로 배열된 퍼즐이 풀수있는지 판별하는 함수
374 {
375     int inv_count = 0, k = 0, temp; //반전된 숫자 배열 세는 변수 등을 선언
376
377     if (x != MDIM - 1 || y != MDIM - 1) //먼저 빈칸의 위치를 맨 아래 맨 오른쪽으로 이동시킨다.
378     {
379         temp = map[MDIM - 1][MDIM - 1];
380         map[MDIM - 1][MDIM - 1] = 0;
381         map[y][x] = temp;
382         y = MDIM - 1;
383         x = MDIM - 1;
384     }
385
386     for (int i = 0; i < MDIM* MDIM - 2; i++) //섞여진 수들을 일렬로 나열시키고 반전된 수의 수를 샌다.
387         for (int j = i + 1; j < MDIM * MDIM - 1; j++)
388             if (*(map[0] + i) > *(map[0] + j))
389                 inv_count++;
390
391     if (inv_count % 2 == 1) //만약 반전된 수의 수가 홀수 개이면 풀 수 없는 퍼즐이며 0을 리턴한다.
392         return 0;
393     else
394         return 1; //짝수 개이면 풀 수 있는 퍼즐이며 1을 리턴한다.
395 }
396
397 //콘솔 화면에서 커서를 특정 위치로 이동
398 void gotoXY(int x, int y)
399 {
400     COORD Pos = { x, y };
401     SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), Pos);
402 }
```

```
Puzzle_Game - Puzzle_game.cpp*
Puzzle_game.cpp*
Puzzle_Game  puzzleGame  getCommand(void)
401     SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), Pos);
402 }
403
404 int getDirectKey()
405 {
406     char key = Non;
407     while (1) { //키보드가 눌릴때만 루프를 탈출한다.
408         if (_kbhit() != 0) //키보드를 눌렀는지 확인함
409         {
410             //특수 키를 눌렀을 때 버퍼에 2Byte가 발생함, 첫번째 값은 224값을 발생하고 두번째 값은 특수키에 따라 다름
411             //특수 키를 확인하기 위해 2번의 getch()함수를 호출해야 함
412             key = _getch();
413             if (key == 224) key = _getch();
414             break;
415         }
416     }
417     return key;
418 }
```

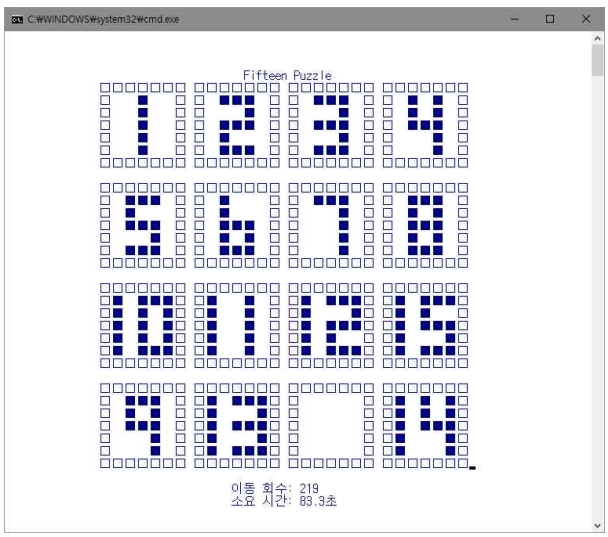
(2) 게임 시작 단계, 게임 도중 단계, 게임 종료 단계의 화면을 이미지 캡쳐하여 프린트 출력하여 제출



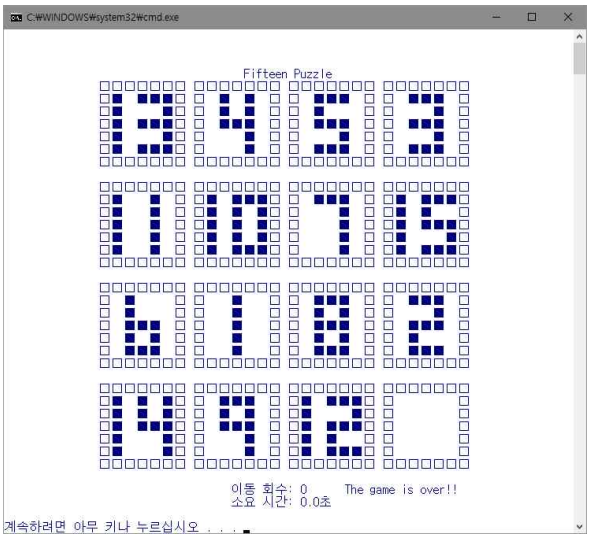
〈섞는 화면〉



〈퍼즐 완성 종료〉



〈게임 진행 중〉



〈게임 강제 종료〉