C++ 팀프로젝트 #2

-C++ 프로그램으로 변환 보고서-

201810754 조윤직

201812617 송재원

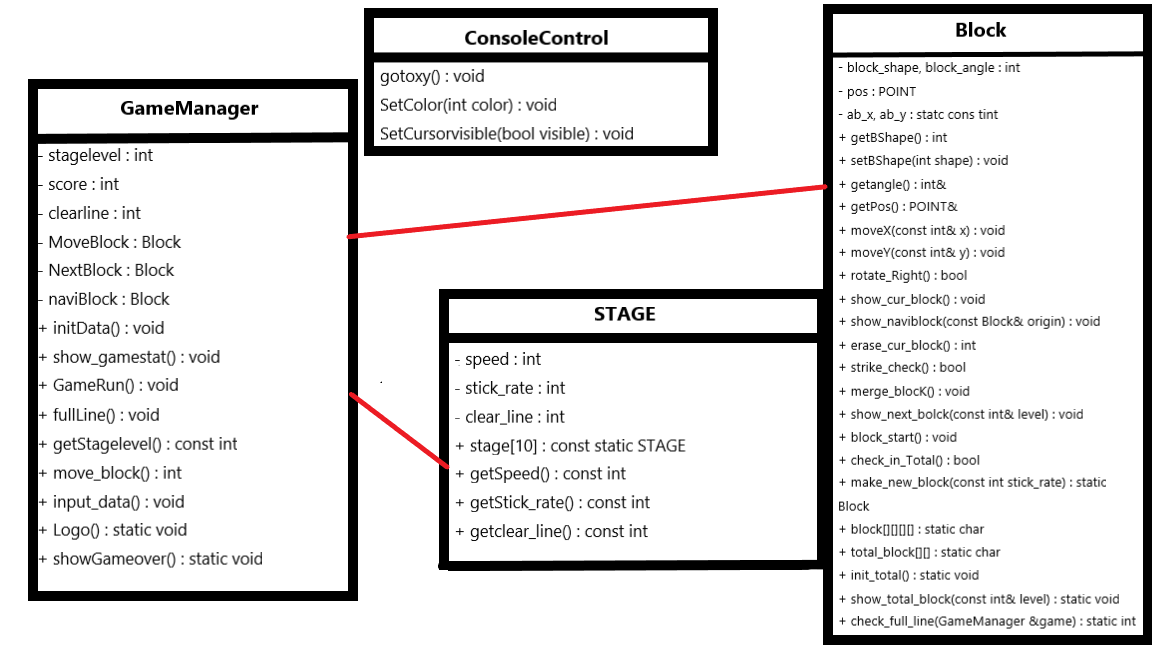
201911175 박진영

201911192 양석준

제출일: 2019. 11.24

**문제 정의 :** #1에서 보완을 한 c 소스를 클래스 구조로 설계하여 c++로 변환하는 문제

**문제 분석 및 해결 :** Printf, getch 등의 c언어 때부터 쓰던 콘솔 입출력 함수를, c++에서 새로 추가된 cin,cout으로 사용하는 바꾸는 것 보다는 CPP의 객체지향언어 특성에 중점을 두어 수정하였다.

**구조 설계 :** 

**각 클래스 -설계 및 소스코드 :**

1**. ConsoleControl 클래스**

(클래스로 선언하지는 않았지만 콘솔창을 컨트롤하는 함수끼리 묶어 두었다.) -콘솔 창 컨트롤 함수 집합

* 1. Gotoxy(int x, int y) 콘솔창의 커서를 x,y위치로 이동해주는 함수:
  2. SetColor(int n): 콘솔창의 출력 색을 지정해주는 함수
  3. SetCursorvisible(bool visible) :(추가적으로 지정한 함수) :콘솔창의 커서를 보이게/안보이게 해주는 함수

#include "ConsoleControl.h"

void gotoxy(int x, int y)

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE); //윈도우 핸들 -콘솔창의 핸들을 변수로 설정

COORD pos; //long 변수x,y 두개를 가지고있는 윈도우api내 구조체 COORD

pos.Y = y;

pos.X = x;

SetConsoleCursorPosition(hConsole, pos); //콘솔 핸들의 커서를 pos위치로 설정

return;

}

void SetColor(int color)

{

HANDLE std\_output\_handle = GetStdHandle(STD\_OUTPUT\_HANDLE); //윈도우 핸들 -콘솔창의 핸들을 변수로 설정

SetConsoleTextAttribute(std\_output\_handle, color); //콘솔핸들의 텍스트 색 설정

}

void SetCursorvisible(bool visible)

{

CONSOLE\_CURSOR\_INFO cursorInfo = { 0, };

cursorInfo.dwSize = 1;

cursorInfo.bVisible = visible;

SetConsoleCursorInfo(GetStdHandle(STD\_OUTPUT\_HANDLE), &cursorInfo);

}

1. **Stage 클래스**

A. 설계

|  |  |
| --- | --- |
| #pragma once  Class STAGE // 스테이지 1~10의 정보(속도, 클리어 할 라인 수, 스틱 나올확률)저장 클래스  {  private:      int speed; //스테이지의 블록떨어지는 속도 담을 변수      int stick\_rate; //스테이지의 블록이 스틱이 될 확률을 담을 변수      int clear\_line; //스테이지 완료 조건을 담을 변수      STAGE(int speed, int stick\_rate, int clear\_line) :speed(speed), stick\_rate(stick\_rate), clear\_line(clear\_line) {        }//생성자를 private화해서 외부에서 stage객체를 생성하지 못하게함      ~STAGE() {}  public:      //상수로 선언 시켜둔 stage의 정보 10개      const static STAGE stage[10];        int getSpeed()const { return speed; }      int getStick\_rate()const { return stick\_rate; }      int getclear\_line()const { return clear\_line; }    };  const STAGE STAGE::stage[10] = { STAGE(40,20,20),STAGE(38,18,20),STAGE(35,18,20),STAGE(30,17,20),STAGE(25,16,20),                  STAGE(20,15,20),STAGE(15,14,20),STAGE(10,13,20),STAGE(6,12,20) ,STAGE(4,11,99999) }; | [cs](http://colorscripter.com/info#e) |

3. **Block 클래스**

A. 설계:

|  |  |
| --- | --- |
| #pragma once  class GameManager;  #include "mainheader.h"    #ifndef DEFBLOCK  #define DEFBLOCK 0  class Block  {  private:      //블록 모양,각도 저장 변수      int block\_shape, block\_angle;      POINT pos; //블록 위치 저장 변수        //블록 절대 위치 - 이곳을 기준으로 블록위치 지정      static const int ab\_x = 5, ab\_y = 1;    public:      Block(); //블록 생성자      Block(int shape, int angle = 0, POINT pos = { 0, 0 });      ~Block();      //멤버 변수 겟 함수      int getBShape();      void setBshape(int shape);      int& getangle();      POINT& getPos();      //x축방향으로 x 만큼  이동해주는 함수      void moveX(const int& x);      //y축 방향으로 y만큼 이동해주는 함수      void moveY(const int& y);      //회전할수 있을 때에만 angle+1 해주는 함수      bool rotate\_Right();          //블록 표시 모양과,각도 ,x,y를 받아서  블록 출력      void show\_cur\_block();      // 네비블록 표시      void show\_naviblock(const Block& origin);      //블록의 모양 각도 위치를 받아 출력 되었었던 블록을 빈 칸으로 그대로 덮어 씌워 지워줌      int erase\_cur\_block();      //충돌 체크 함수 -매개 x      bool strike\_check();      //자기자신블록과 토탈 블록을 토탈블록에 병합한다. 매개x      void merge\_block();      //다음 블록 출력함수-다음블록 칸에 출력하기      void show\_next\_block(const int& level);      //블록 이동 시작전 위치 초기화 함수      void block\_start();      //지금 블록이 게임 화면상에 있는지 확인해주는 함수      bool check\_in\_Total();        //새로운 블록 생성 새로운 블록의 모양 설정 ,- 레벨에따른 스틱 확률에 따른..      static Block make\_new\_block(const int stick\_rate);        //shape,각도별 블록모양-      static const char block[7][4][4][4];        //전체 블록      static char total\_block[BOTTOM+1][RIGHTSIDE+1];      //전체 블록 초기화      static void init\_total();      //전체 블록 출력      static void show\_total\_block(const int& level);      //토탈블록의 줄 전체가 가득 찬 부분이 있으면- 제거하고 토탈 블록 데이터 수정      static int check\_full\_line(GameManager &game);      };  #endif // !DEFBLOCK | [cs](http://colorscripter.com/info#e) |

B. 소스 코드 (정의 부)

|  |  |
| --- | --- |
| #include "mainheader.h"    char Block::total\_block[21][14] = { 0, };  const char Block::block[7][4][4][4] = { //[도형의 모양 0~6][도형의 각도 0~3][y][x]     //막대모양     1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,   1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,   1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,   1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,       //네모모양     1,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,   1,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,   1,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,   1,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,       //'ㅓ' 모양     0,1,0,0,1,1,0,0,0,1,0,0,0,0,0,0,   1,1,1,0,0,1,0,0,0,0,0,0,0,0,0,0,   1,0,0,0,1,1,0,0,1,0,0,0,0,0,0,0,   0,1,0,0,1,1,1,0,0,0,0,0,0,0,0,0,       //'ㄱ'모양     1,1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,   1,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,   1,0,0,0,1,0,0,0,1,1,0,0,0,0,0,0,   0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,       //'ㄴ' 모양     1,1,0,0,1,0,0,0,1,0,0,0,0,0,0,0,   1,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,   0,1,0,0,0,1,0,0,1,1,0,0,0,0,0,0,   1,1,1,0,0,0,1,0,0,0,0,0,0,0,0,0,       //'Z' 모양     1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,   0,1,0,0,1,1,0,0,1,0,0,0,0,0,0,0,   1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,   0,1,0,0,1,1,0,0,1,0,0,0,0,0,0,0,       //'S' 모양     0,1,1,0,1,1,0,0,0,0,0,0,0,0,0,0,   1,0,0,0,1,1,0,0,0,1,0,0,0,0,0,0,   0,1,1,0,1,1,0,0,0,0,0,0,0,0,0,0,   1,0,0,0,1,1,0,0,0,1,0,0,0,0,0,0    };      void Block::init\_total()  {      for (int i = TOP; i < BOTTOM; i++)//y좌표      {          for (int j = LEFTSIDE; j <= RIGHTSIDE; j++) //x좌표          {              if ((j == LEFTSIDE) || (j == RIGHTSIDE)) //x좌표가 0또는 13일때 - 왼쪽 오른쪽 벽 블록 설정              {                  total\_block[i][j] = 1; //블록              }              else {                  total\_block[i][j] = 0;   //아니면 빈블록              }          }      }        for (int j = LEFTSIDE; j <= RIGHTSIDE; j++)         //화면의 제일 밑의 줄은 1로 채운다. 맨밑줄          total\_block[BOTTOM][j] = 1;    }    void Block::show\_total\_block(const int& level)  {      SetColor(DARK\_GRAY);// 전체 블록, 색 설정 다크그레이      for (int i = TOP; i <= BOTTOM; i++) //y축      {          for (int j = LEFTSIDE; j <= RIGHTSIDE; j++)//x축          {              if (j == LEFTSIDE || j == RIGHTSIDE || i == BOTTOM)      //레벨에 따라 외벽 색이 변함              {                  SetColor((level % 6) + 1); //레벨에 따른 외벽색                }              else {                  SetColor(DARK\_GRAY); //외벽이 아니면 그냥 다그색으로              }              gotoxy((j \* 2) + ab\_x, i + ab\_y); //해당 블록              if (total\_block[i][j] == 1)//빈칸이 아니면 네모 출력,              {                  printf("■");              }              else {                  printf("  ");              }            }      }      SetColor(BLACK);// 출력하는 글자 색 초기화-      gotoxy(77, 23);//커서 위치 재설정      return;  }    int Block::check\_full\_line(GameManager& game)  {      for (int i = TOP; i < BOTTOM; i++) //전체 모든 줄 체크      {          int j;          for (j = LEFTSIDE + 1; j < RIGHTSIDE; j++) //i 줄의 모든 칸을 체크          {              if (total\_block[i][j] == 0)//중간에 빈 곳이 있으면 break                  break;          }          if (j == 13)   //한줄이 다 채워졌음- j=13 break 없이 반복이 13번 수행된          {                  //출력-동작              show\_total\_block(game.getStagelevel()); //토탈블록 출력              SetColor(BLUE); // 블루로 출력              gotoxy(1 \* 2 + ab\_x, i + ab\_y); // 이 줄 위치로 가서              for (j = 1; j < 13; j++)              {                  printf("□"); //블루색으로 한번 출력 해주고                  Sleep(10); //0.01초마다 출력              }              gotoxy(1 \* 2 + ab\_x, i + ab\_y); //다시 줄의 시작으로 가서              for (j = 1; j < 13; j++)              {                  printf("  ");// 쭉 지워줌                  Sleep(10);              }                //데이터 수정부              for (int k = i; k > 0; k--)//지금 줄 i부터 그 윗줄 반복              {                  for (j = 1; j < RIGHTSIDE; j++) //x축                      total\_block[k][j] = total\_block[k - 1][j];//바로 위에 줄의 정보를 가져옴              }                for (j = 1; j < 13; j++) //맨 위의 줄 0으로 초기화 해줌                  total\_block[0][j] = 0;              game.fullLine();            }      }      show\_total\_block(game.getStagelevel());      return 0;  }          Block::Block()  {  }    Block::Block(int shape, int angle, POINT pos) :block\_shape(shape), block\_angle(angle), pos(pos)  {  }    Block::~Block()  {  }    int Block::getBShape()  {      return block\_shape;  }    void Block::setBshape(int shape)  {      this->block\_shape = shape;  }    int& Block::getangle()  {      return block\_angle;  }        POINT& Block::getPos()  {      return pos;  }    void Block::moveX(const int& x)  {      pos.x += x;  }    void Block::moveY(const int& y)  {      pos.y += y;  }    bool Block::rotate\_Right()  {      block\_angle = (block\_angle + 1) % 4; //angle을 하나 올리고      if (strike\_check()) { //충돌하면          block\_angle = (block\_angle - 1) % 4; //복귀          return false; //성공 실패리턴이나 결국 사용하지 않음      }      else          return true; //  }        Block  Block::make\_new\_block(const int stick\_rate)  {      // i= 확률      int i;      i = rand() % 100;      if (i <= stick\_rate)      //막대기 나올확률 계산          return Block(0);                //막대기 모양으로 리턴      else          return Block((rand() % 6) + 1);    }    void Block::show\_cur\_block()  {      //정의 해둔 SetColor함수를 이용하여 모양에 따른 색 출력 구현      switch (block\_shape)      {      case 0:          SetColor(RED);          break;      case 1:          SetColor(BLUE);          break;      case 2:          SetColor(SKY\_BLUE);          break;      case 3:          SetColor(WHITE);          break;      case 4:          SetColor(YELLOW);          break;      case 5:          SetColor(VOILET);          break;      case 6:          SetColor(GREEN);//greenjoa!          break;      default:          return;      }      for (int i = 0; i < 4; i++)      {          for (int j = 0; j < 4; j++)          {              if ((j + pos.y) < 0)// 출력되는 y좌표가 0이하 콘솔 창밖을 벗어날때는 출력하지 않음                  continue;                if (block[block\_shape][block\_angle][j][i] == 1) //해당하는 도형의 j,i위치가 빈블록이 아닐 때              {                  gotoxy((i + pos.x) \* 2 + ab\_x, j + pos.y + ab\_y); //(i+x)\*2 + ab\_x,j+y+ab\_y 위치에 출력 저 사각형의 출력이 2칸 차지하므로 x는 두배                  printf("■");                }          }      }        SetColor(BLACK);// 출력하는 글자 색 초기화-      gotoxy(77, 23);//커서 위치 재설정      return;  }    //네비블록 출력 함수  void Block::show\_naviblock(const Block& origin)  {      this->erase\_cur\_block(); //이전 네비 블록을 지우고      \*this = origin;      //origin으로 받아온 현재 이동블록으로 자기자신초기화        switch (block\_shape)      {//shape에 따라서 ..색 출력      case 0:          SetColor(DARK\_RED);          break;      case 1:          SetColor(DARK\_BLUE);          break;      case 2:          SetColor(DARK\_SKY\_BLUE);          break;      case 3:          SetColor(GRAY);          break;      case 4:          SetColor(DARK\_YELLOW);          break;      case 5:          SetColor(DARK\_VOILET);          break;      case 6:          SetColor(DARK\_GREEN);          break;      }          while (strike\_check() != true) //충돌 할때까지          moveY(1); //내려감      moveY(-1);        //출력부      for (int i = 0; i < 4; i++)//y축      {          for (int j = 0; j < 4; j++) //x축          {              if ((j + pos.y) < 0) //0보다 작을때에는 출력x                  continue;                if (block[block\_shape][block\_angle][j][i] == 1)              {                  gotoxy((i + pos.x) \* 2 + ab\_x, (j + pos.y) + ab\_y);                  printf("■");                }          }      }        //출력 후 초기화      SetColor(BLACK);      gotoxy(77, 23);      return;  }    //블록 지우기 함수  int Block::erase\_cur\_block()  {        if (block\_shape <= 6 && block\_shape >= 0)          for (int i = 0; i < 4; i++) //y축          {              for (int j = 0; j < 4; j++) //x축              {                  if (block[block\_shape][block\_angle][j][i] == 1)//해당하는 블록위치가 빈 블록이 아닐 때                  {                      if (total\_block[j + pos.y][i + pos.x] == 1) break; //이동블록이 끝까지 이동한후 다음블록이 끝까지 이동할때까지 출력되지 않는 문제가 있어 토탈블록의 부분일때는 안지워주게 추가                      gotoxy((i + pos.x) \* 2 + ab\_x, j + pos.y + ab\_y);//이 위치에 빈블록으로 지움                      printf("  ");                      //break;                    }              }          }        return 0;  }    //충돌 체크  bool Block::strike\_check()  {      int block\_dat;        if (block\_shape <= 6 && block\_shape >= 0) {          for (int i = 0; i < 4; i++)//y          {              for (int j = 0; j < 4; j++)//x              {                  if ((pos.x + j) == LEFTSIDE || (pos.x + j) == RIGHTSIDE)                      block\_dat = 1;   //이 블록이 지금 토탈 블록의 외곽부 이면 -1                  else if (pos.y + i < 0) //이경우 total\_block  에는 이부분이 없기에 추가  수정--                      block\_dat = 0;                  else                      block\_dat = total\_block[pos.y + i][pos.x + j];  //이블록이 토탈블록에 빈블록인지,찬 블록인지                      if ((block\_dat != 0) && (block[block\_shape][block\_angle][i][j] == 1))                  { //좌측벽의 좌표를 빼기위함? -- 이블록이 빈블록이 아니고 도형의 이 블록 부분도 1이면 충돌 !                      return true;                  }              }          }          return 0;      }      return 1;    }    //블록 병합  void Block::merge\_block()  {      for (int i = 0; i < 4; i++)//y축      {          for (int j = 0; j < 4; j++)//x축          { //토탈 블록에 현재블록 부분을 1로 바꿔줌              total\_block[pos.y + i][pos.x + j] |= block[block\_shape][block\_angle][i][j];          }      }      return;    }    void Block::show\_next\_block(const int& level)  {        SetColor((level + 1) % 6 + 1); //다음블록이 표기되는 테두리는 레벨따라 색이 다름      for (int i = 1; i < 7; i++) //다음에 무슨 블록이 나올지 알려주는 네모난 공간을 만듦      {          gotoxy(33, i);          for (int j = 0; j < 6; j++)          {              // 첫 행과 마지막 행만 색칠된 네모칸을 쭉 출력함. 나머지는 첫 번째 열과 마지막 줄 열에만 네모 출력.              if (i == 1 || i == 6 || j == 0 || j == 5)              {                  printf("■");              }              else {                  printf("  ");              }            }      }      Block temp(block\_shape, 0, { 15, 1 });      temp.show\_cur\_block(); // 15\*2+ab\_x하면 35,  다음블록 안에다가 도형 출력  }    void Block::block\_start()  {      pos = { 5,-3 };      block\_angle = 0;  }    bool Block::check\_in\_Total()  {      if (block\_shape <= 6 && block\_shape >= 0) {          for (int i = 0; i < 4; i++)//y for문 순서상 가장 위에잇는 칸부터          {              for (int j = 0; j < 4; j++)//x              {                  if (block[block\_shape][block\_angle][i][j] == 1) { //가장위에 있는 블록                      if (pos.y + i < 0) //블록의 가장위쪽 위치가 0보다 작으면 밖에 있음!                          return false;                      else                          return true;//아니면 안에                  }              }          }      }  } | [cs](http://colorscripter.com/info#e) |

1. GameManager 클래스
   1. 설계 :

|  |  |
| --- | --- |
| #pragma once  #include "mainheader.h"  #include "STAGE.h"  #ifndef DEFGAME  #define DEFGAME 0  class GameManager  {  private:      //현재 스테이지 레벨      int stagelevel;      //현재 점수      int score;      //현재 클리어한 라인      int clearline;        //현재 이동하는 블록      Block MoveBlock;      //다음 블록      Block NextBlock;      //그림자 블록      Block naviBlock;    public:      GameManager();      ~GameManager();      //정의된 check\_full\_line함수  토탈블록이 변화 했으니 줄이 완성 되었으면 지워주기 위해      void initData();      //게임 점수판 출력 함수 -해당 위치에 스테이지,점수 라인 정보 출력      void show\_gamestat();      //게임 주 실행함수      void GameRun();      //한 줄이 다찼으면 , 점수,clearline, 점수창 수정      void fullLine();      // 현재 스테이지 리턴 함수      int getStagelevel() const;        //현재 이동블록을 움직여주는 함수      // 리턴 값- 0은 정상 한칸 이동, 1이 리턴 게임 오버,      //2가 리턴 내려올 수 있는 곳까지 내려왔음(게임오버x)-이 경우 지금블록 병합처리 ,다음블록 처리      int move\_block();      //게임 시작시 - 도움말표기, 스테이지 선택 함수      void input\_data();        //객체에 따라 다른 특성이 없는 함수는 정적으로      static void Logo(); //게임 로고출력함수      static void showGameover();  };  #endif // !DEFGAME | [cs](http://colorscripter.com/info#e) |

* 1. 소스 코드 (정의 부)

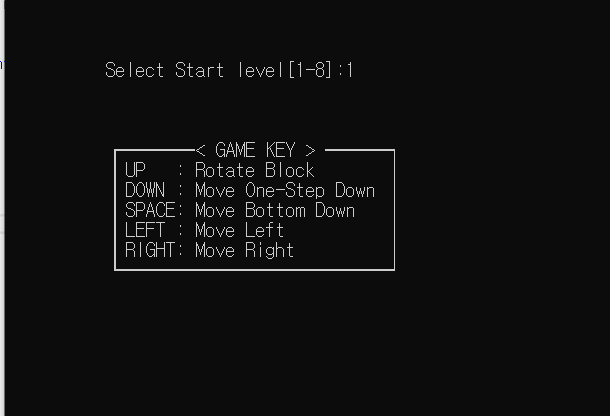
|  |  |
| --- | --- |
| #include "mainheader.h"    GameManager::GameManager()  {      initData();  }    GameManager::~GameManager()  {    //데이터 초기화 함수  void GameManager::initData()  {      //블록의 토탈 블록 초기화함수 실행      Block::init\_total();      //현재 스테이지레벨, 라인수,점수 초기화      stagelevel = 0;      clearline = 0;      score = 0;  }    void GameManager::show\_gamestat()  {      //처음에 한번만 출력한뒤 스테이지 스코어 라인 글씨는 다시 출력 안하려 한 부분인거 같은데  //printed\_text를 0이 아닌 수로 변하는 부분이 없어 딱히 의미가 없는듯? -일단 그냥 유지      static int printed\_text = 0; // 밑에있는 조건문의 조건에 들어가는 변수 조건문을 실행해줌        SetColor(GRAY);      if (printed\_text == 0)      { //점수 출력          gotoxy(35, 7);          printf("STAGE");            gotoxy(35, 9);          printf("SCORE");            gotoxy(35, 12);          printf("LINES");          }      gotoxy(41, 7);      printf("%d", stagelevel + 1);      gotoxy(35, 10);      printf("%10d", score);      gotoxy(35, 13);      printf("%10d", STAGE::stage[stagelevel].getclear\_line() - clearline);  }    void GameManager::GameRun()  {      //게임 오버 여부 담을 변수      SetCursorvisible(false); //시작시 커서끔      int is\_gameover = 0;      initData(); //초기화 함수 수행      Logo(); //로고 띄우고      while (true) //      {          SetCursorvisible(true);//스테이지 선택시 커서 보여줌          input\_data(); //스테이지 선택 함수          SetCursorvisible(false); //스테이지 선택후 커서 끔          Block::show\_total\_block(stagelevel); //토탈 블록 출력 - 맨처음 프레임 출력            //최초 블록 세팅          MoveBlock = Block::make\_new\_block(STAGE::stage[stagelevel].getStick\_rate()); //블록 생성          MoveBlock.block\_start(); //시작 블록 초기화            NextBlock = Block::make\_new\_block(STAGE::stage[stagelevel].getStick\_rate()); //다음 블록 생성            naviBlock.show\_naviblock(MoveBlock); //네비 블록보이기          MoveBlock.show\_cur\_block(); // 이동블록 출력-네비보다 먼저 하면 네비에서출력이지워짐          NextBlock.show\_next\_block(stagelevel); //다음 블록 칸 표시          show\_gamestat();            for (int i = 0; true;  i++) { //무한 루프              if (\_kbhit()) { //키입력이 있으면                  char keytemp = \_getch();                  if (keytemp == KEY\_ESC) {        //esc입력이 들어오면-종료                      initData(); //데이터 초기화                      system("cls"); //전체 초기화                      MoveBlock.block\_start(); //                      break;                  }                  if (keytemp == EXT\_KEY) {//확장키- 방향키입력시                      keytemp = \_getch();//확장키 구분                      switch (keytemp)                      {                      case KEY\_UP: { //업키- 회전 키                          MoveBlock.erase\_cur\_block(); //이동블록 일단 지우고                          MoveBlock.rotate\_Right(); //회전 시킬수 있을 때에만 회전해주는 함수                          naviBlock.show\_naviblock(MoveBlock); //네비 블록보이기                          MoveBlock.show\_cur\_block(); // 이동블록 출력                          break;                      }                      case KEY\_LEFT: {//좌측키 -왼쪽이동키                          if (MoveBlock.getPos().x > LEFTSIDE) { //이동블록이 가장 좌측에 있지 않으면                              MoveBlock.erase\_cur\_block(); //이동 블록 일단지우고                              MoveBlock.moveX(-1); //한칸 이동후                              if (MoveBlock.strike\_check()) { //충돌하면                                  MoveBlock.moveX(1); //다시 원 위치로                              }                              naviBlock.show\_naviblock(MoveBlock); //네비 블록보이기                              MoveBlock.show\_cur\_block(); // 이동블록 출력                          }                          break;                      }                      case KEY\_RIGHT: {//우측키 오른쪽 이동키                          if (MoveBlock.getPos().x < RIGHTSIDE) {//이동 블록이 가장 우측에 있지 않으면 -이거는 별로의미가 없는듯                              MoveBlock.erase\_cur\_block(); //이동 블록 일단 지우고                              MoveBlock.moveX(1); //한 칸 이동후                              if (MoveBlock.strike\_check()) {//충돌하면                                  MoveBlock.moveX(-1); //다시 원위치로                              }                              naviBlock.show\_naviblock(MoveBlock); //네비 블록보이기                              MoveBlock.show\_cur\_block(); // 이동블록 출력                          }                          break;                      }                      case KEY\_DOWN: { //아래키 - 내려가게하는 키                          is\_gameover = move\_block(); //블록을 한칸 아래로 이동함수                          if(is\_gameover==0)  //0값이 출력 되었으면                              naviBlock.show\_naviblock(MoveBlock); //네비 블록보이기                          //is\_gameover ==2인 아래에 도달했을때는 어쩌피 네비와 위치가 같아 출력x                          MoveBlock.show\_cur\_block(); // 이동블록 출력                            break;                      }                      default:                          break;                      }                      continue;                  }                  if (keytemp == SPACEBAR) { //스페이스바를 누르면                      while (is\_gameover == 0) {                          is\_gameover = move\_block();// 블록이 아래에 도착할 때까지 수행                      }//이동블록 출력  이부분에서는 네비블록과 위치가 동일 하기에..                      MoveBlock.show\_cur\_block();                  }                }              if (is\_gameover == 1) { //게임 오버되면                  showGameover(); //게임오버 출력                  SetColor(GRAY); //                  initData(); //데이터 초기화                  MoveBlock.block\_start();                  is\_gameover = 0;                  break;              }              if (STAGE::stage[stagelevel].getclear\_line() <=clearline) //깬 라인수가 클리어 수보다 많으면              {                  clearline -= STAGE::stage[stagelevel].getclear\_line(); //라인수 초기화                  Block::show\_total\_block(++stagelevel);  //줄 초기화 -수정                  show\_gamestat();                }              if ((i % STAGE::stage[stagelevel].getSpeed()) == 0) {                  if (is\_gameover != 1) {                      is\_gameover = move\_block();                      naviBlock.show\_naviblock(MoveBlock); //네비 블록보이기                      MoveBlock.show\_cur\_block(); // 이동블록 출력                  }                }              //출력부 초기화              gotoxy(77, 23);              Sleep(15);               gotoxy(77, 23);          }          //함수 초기화          initData();      }    }    void GameManager::fullLine()  {      clearline += 1; //라인 클리어 되면 라인 수+1      score += 100 + (stagelevel \* 10) + (rand() % 10); //점수 레벨에따른 상승      show\_gamestat(); //게임 정수판 수정  }    int GameManager::getStagelevel() const  {      return stagelevel;  }  //블록움직여 주는 함수  int GameManager::move\_block()  {      MoveBlock.erase\_cur\_block(); //이전 위치의 블록 지우기        MoveBlock.moveY(1); //아래로 한 칸 이동      if (MoveBlock.strike\_check()) {// 만약 이블록이 토탈 블록과 충돌 했으면          MoveBlock.moveY(-1);          if (MoveBlock.check\_in\_Total()) {//블록 다 떨어짐                MoveBlock.merge\_block();              Block::check\_full\_line(\*this);              Block::show\_total\_block(stagelevel);              show\_gamestat(); //게임 정수판 수정              MoveBlock = NextBlock; //블록 새롭게 바꾸기              MoveBlock.block\_start(); //              NextBlock = Block::make\_new\_block(STAGE::stage[stagelevel].getStick\_rate());              NextBlock.show\_next\_block(stagelevel);              return 2;          }          else {              return 1;          }        }      return 0;  }    //로고 출력 함수  void GameManager::Logo()  {      int i, j;// 반복문에서 사용되는 변수      //이 출력부도 수정      gotoxy(13, 3);      printf("┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓"); //수정      Sleep(100);      gotoxy(13, 4);      printf("┃◆◆◆  ◆◆◆  ◆◆◆   ◆◆     ◆   ◆  ◆ ┃");      Sleep(100);      gotoxy(13, 5);      printf("┃  ◆    ◆        ◆     ◆ ◆    ◆    ◆◆  ┃");      Sleep(100);      gotoxy(13, 6);      printf("┃  ◆    ◆◆◆    ◆     ◆◆     ◆     ◆   ┃");      Sleep(100);      gotoxy(13, 7);      printf("┃  ◆    ◆        ◆     ◆ ◆    ◆    ◆◆  ┃");      Sleep(100);      gotoxy(13, 8);      printf("┃  ◆    ◆◆◆    ◆     ◆  ◆   ◆   ◆  ◆ ┃");      Sleep(100);      gotoxy(13, 9);      printf("┗━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┛"); //수정        gotoxy(28, 20);      printf("Please Press Any Key~!");      // 키를 입력 받을때 까지 계속 반복해서 블록들 출력함      for (i = 0; i >= 0; i++)      {          if (i % 40 == 0)          {                for (int i = 14; i < 19; i++) {                  gotoxy(6, i); printf("%95s", ""); //수정-잔상지우기위해              }              Block(rand() % 7, rand() % 4, {6,14}).show\_cur\_block();              Block(rand() % 7, rand() % 4, { 12,14 }).show\_cur\_block();              Block(rand() % 7, rand() % 4, { 19,14 }).show\_cur\_block();              Block(rand() % 7, rand() % 4, { 24,14 }).show\_cur\_block();            }          if (kbhit())              break;          Sleep(30);      }        getche();      system("cls");        return;  }    //스테이지 선택 함수  void GameManager::input\_data()  {      int i = 0; // 입력받는 값을 변수로 선언      SetColor(GRAY);      gotoxy(10, 7);      printf("┏━━━━━━━ < GAME KEY > ━━━━━━┓"); //수정      Sleep(10);      gotoxy(10, 8);      printf("┃ UP   : Rotate Block       ┃");      Sleep(10);      gotoxy(10, 9);      printf("┃ DOWN : Move One-Step Down ┃");      Sleep(10);      gotoxy(10, 10);      printf("┃ SPACE: Move Bottom Down   ┃");      Sleep(10);      gotoxy(10, 11);      printf("┃ LEFT : Move Left          ┃");      Sleep(10);      gotoxy(10, 12);      printf("┃ RIGHT: Move Right         ┃");      Sleep(10);      gotoxy(10, 13);      printf("┗━━━━━━━━━━━━━━━━━━━━━━━━━━━┛"); //수정          while (i < 1 || i>8)// 1미만 8초과인수일때 다시 입력받음      {          gotoxy(10, 3);          printf("Select Start level[1-8]:              \b\b\b\b\b\b\b\b\b\b\b\b\b\b");          scanf("%d", &i);          while (getchar() != '\n'); //버퍼를 비움:scanf의 경우 엔터가 버퍼에 남아 값이 있다고 인식되어 무한반복되므로          /\*          if (scanf("%d", &i) == 0) { 위의 부분을 이런 식으로 바꿔도 ㄱㅊ             rewind(stdin);          }          \*/        }          stagelevel = i - 1;      system("cls");      return;  }    //게임오버 출력  void GameManager::showGameover()  {      //게임 오버 출력      SetColor(RED);      gotoxy(15, 8);      printf("┏━━━━━━━━━━━━━━━━━━━━━━━━━━┓");      gotoxy(15, 9);      printf("┃\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*┃");      gotoxy(15, 10);      printf("┃\*        GAME OVER       \*┃");      gotoxy(15, 11);      printf("┃\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*┃");      gotoxy(15, 12);      printf("┗━━━━━━━━━━━━━━━━━━━━━━━━━━┛");      fflush(stdin); //지금까지 쌓여있던 입력 버퍼를 다 지워줌      Sleep(1000); //1초 정지후        getche();      system("cls");        return;  } | [cs](http://colorscripter.com/info#e) |

**실행 결과 :**

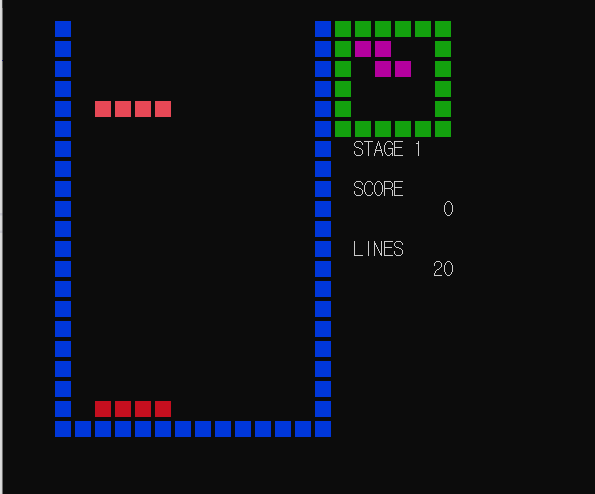
**(1) 타이틀 화면**



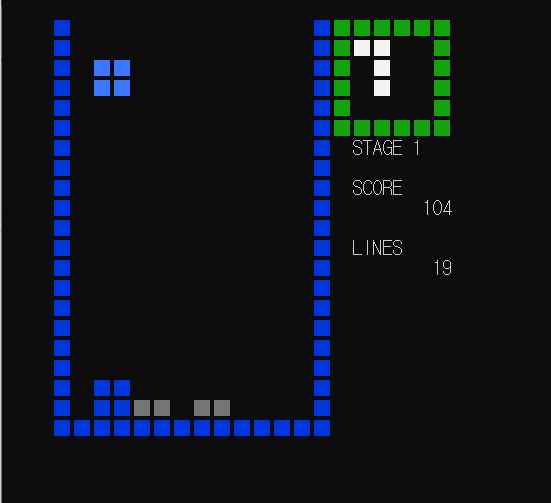
**(2) 레벨 선택 화면**



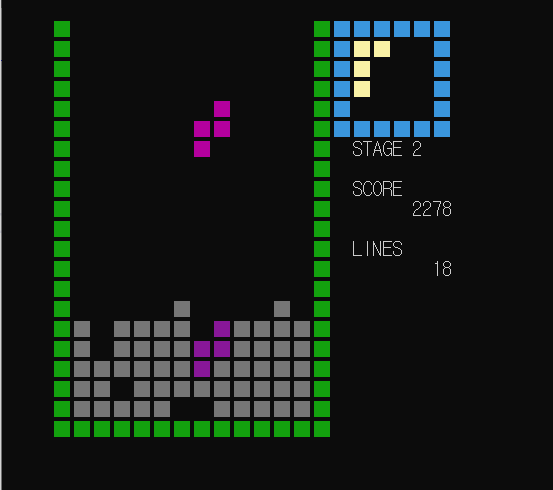
**(3) 블록 그림자 기능 추가**



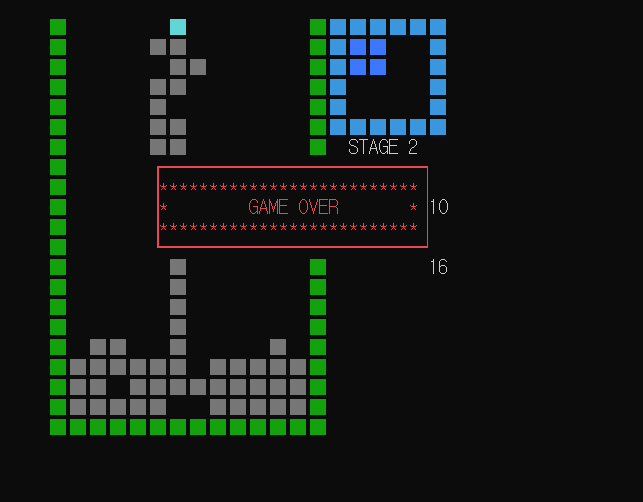
**(4) 한 줄 지우기**

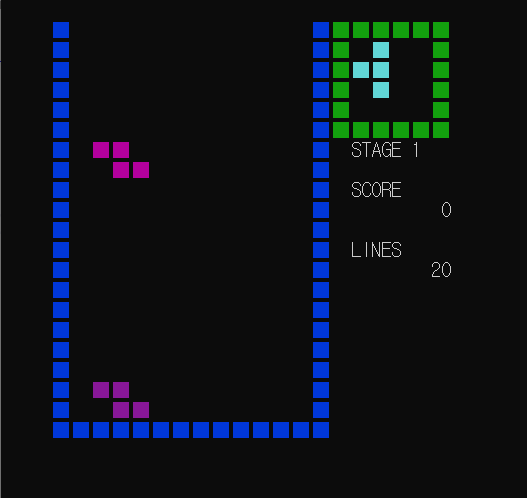


**(5) 스테이지 클리어**



**(6) 게임 오버**





**느낀점 및 토의사항**

* **C로 구현되어 있는 소스를 C++로 변환하면서 클래스를 설계화하는 과정이 굉장히 어려웠다. 이를 해결하기 위하여, 팀원들과 구조를 먼저 그리는 데 시간을 오래 투자하였다.**
* **이번 변환 과정에서 가장 어려웠던 부분은 main 소스에 선언되어 있던 함수들을 클래스에 집어넣는 과정이었다. 이와 더불어 이를 조원과 역할을 나누고 합치는 과정 역시 힘든 과정이었다. 하지만 이 과정이 어려웠기에 이를 조원들과 함께 극복해나가는 과정이 매우 뜻깊었다고 생각한다.**
* **C++ 프로그램 변환을 통해서 한번에 쓰여진 소스코드가 다양하게 객체지향 속성을 이용해 여러 클래스로 분할 설계하여 프로그램을 다시 짜보았다. 처음에 다양한 변수들을 어떻게 호출, 저장할까 하는 고민과 팀원과 의사소통이 중요함을 알았다. 추가기능도 넣어봤는데 잘 구현한 기능인 것 같다. 추가적으로 앞으로 어떤 스토리를 구상하여 프로그램을 만들기 생각해 보았다.**