C++ 팀프로젝트 #1

-분석 및 버그 수정 보고서-

201810754 조윤직

201812617 송재원

201911175 박진영

201911192 양석준

제출일: 2019. 11. 15

**문제정의 :** tetris.cpp 단일 소스로 구성된 프로그램을 분석하고 버그를 수정

**찾은 버그 목록:**

* **Windows.h의 부재:**
  + **문제 : gotoxy,Setcolor 함수, 등에서 윈도우api를 사용하나 윈도우 api를 포함하고 있는 헤더가 정의 되지 않아 오류가 발 생**
  + **해결 : 위쪽 해더 추가 부분에 #include <Windos.h>를 추가**
* **time.h 의 부재:**
  + **문제 : 랜덤 변수를 설정 할 때 시드 값으로 시간을 주는데 이 시간을 가져오기 위해 정의되어야 하는 time.h가 사용되지 않음**
  + **해결 : 헤더부에 #includ <time.h> 를 추가**
* **#pragma warning:**
  + **문제 : scanf, getch 등의 함수의 경우 특정 버전부터 보안상의 이유로 사용이 막아져 있어 scanf,getch 함수를 사용 할 수 없다.**
  + **해결 : 프로그램에서 사용되는 입력 함수가 문자열을 받는 경우가 없으니 scanf\_s, 등의 다른 함수를 사용하지 않고 전처리기에서 사용을 못하게하는 부분을 처리해준다 헤더 부에 #progma warning(disable:4996) 추가**
* **게임로고 박스,게임오버 박스 ,게임 키 도움말 박스 출력수정**
  + **문제 : 게임로고,게임 오버 박스, 게임 도움말 박스가 출력 부분에서 크기가 맞지 않음**
  + **해결 : 박스가 서로 크기가 맞게 끔 출력 수정**
* **게임 로고 부분 테트리스 블록 잔상:**
  + **문제 : 게임 로고에서 테트리스 블록이 계속 변하면서 출력 되는데 이전에 출력되는 블록이 새로운 블록의 영역에서 벗어나는 경우 잔상이 남는 문제가 존재**
  + **해결 : 출력 시 마다 테트리스 블록이 출력되는 부분 전체를 지우고 다시 출력하게 -show\_logo함수를 변경 하였다.**
* **Strike\_check부분에 -블록이 전체 블록 밖에 있을 때 부분 추가**
  + **문제 : totalblock은 [0~20][0~13]에서만 정의 되지만 strikecheck실행 부에는 [음수][0~13]인 경우가 포함되어 수정필요**
  + **해결 : strike\_check함수 수정 – 자세한 내용 함수 분석에 포함**
* **가끔 게임 종료 시 엑세스 에러 발생:**
  + **문제 : shape나 y,값이 이상하게 변경되어 종료 과정 중 엑세스 에러 발생**
  + **해결 : shape나 y값이 이상하게 변하는 곳을 찾아서 수정을 해야하나 찾아내지 못하여 strike\_check, show\_cur\_block, erase\_cur\_block 실행중 shape 0~6을 벗어나면 함수의 shape가 실행되는 실행부를 실행하지 않게 변경**
* **Esc를 누르면서 게임하면 잔상이 남음**
  + **문제: esc를 눌러주면서 게임을 하면 블록이 이동하면서 지워져야 하지만 일부분이 남으면서 전체가 다 지워지지 않음**
  + **해결 : 소스 코드상의 esc의 코드 27을 사용하는 부분이 없음에도 불구하고 오류가 발생하는 것으로 해결방안을 찾지 못해 esc를 누르면 스테이지를 종료하고 스테이지 선택창으로 돌아가게 변경**
* **점수 초기화 되지 않음**
  + **문제 : 게임 오버 되었을 때 점수가 초기화 되지 않음**
  + **해결 : init함수 부분에 점수 초기화 부분을 추가**
* **라인이 한 개 남았을 때 2개이상의 라인을 깨면 영원히 클리어 되지 않음**
  + **문제 : 라인이 한 개 남았을 때 2개이상의 라인을 부술 경우 게임이 끝나지 않음**
  + **해결 : main의 다음레벨이 넘어가는 조건을 ==에서 <=로 수정**

**-프로그램 분석:**

**전역 선언부**

|  |  |
| --- | --- |
| #include <stdio.h> //c언어 기본 라이브러리  #include <conio.h> // visual에서 제공하는 콘솔 입출력 라이브러리 -getche 의 사용을 위해  #include <Windows.h> //윈도우 api제공 헤더  #include <time.h> //랜덤 변수의 시드를 시간으로 해주기 위해  //#include <string.h>//string.h헤더 부분을 사용하는 부분이 없어 주석처리  #pragma warning(disable: 4996) //scanf에러 방지  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //상수 선언  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  #define EXT\_KEY         0xffffffe0   //확장키 인식값  #define KEY\_LEFT      0x4b  #define KEY\_RIGHT      0x4d  #define KEY\_UP         0x48  #define KEY\_DOWN      0x50  #define KEY\_ESC             0x1b   //ESC의 인식값 -esc의 종료를 위해 추가해준 정의  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*    //구조체 선언  struct STAGE {      //각 스테이지마다의 난이도 설정      int   speed;   //숫자가 낮을수록 속도가 빠르다      int stick\_rate;      //막대가 나오는 확률 0~99 , 99면 막대기만 나옴      int clear\_line;  // 이 스테이지에서 클리어 해야할 라인수  };    enum {...} //열거형 SetColor 함수에서 사용을 위한 - 특정 컬러의 번호 0~15에 대한 열거형 정의 | [cs](http://colorscripter.com/info#e) |

**전역변수**

|  |  |
| --- | --- |
| int level; // 현재 스테이지 변수 --각 함수에서 레벨에 따라 달라지는 값이 있어 전역변수로 선언  int ab\_x, ab\_y;   //화면중 블럭이 나타나는 좌표의 절대위치    //각각 현재 블록(지금 이동중인 블록)의 모양, 각도, 현재 위치x,y 를 나타내주는 변수  int block\_shape, block\_angle, block\_x, block\_y;    int next\_block\_shape; //다음블록의 모양을 담는 변수  int score; //현재 점수 계산 함수  int lines; //이 스테이지에서 완료 한 줄 수    //화면에 표시되는 블럭들 전체프레임+ 쌓인 블록  char total\_block[21][14];  //1~10(0~9) 스테이지의 정보 배열  struct STAGE stage\_data[10];  char block[7][4][4][4] = {...} // 각 도형의 모양정의[도형의 모양 0~6][도형의 각도 0~3][y][x] | [cs](http://colorscripter.com/info#e) |

**함수**

**1.gotoxy함수: 커서를 이동하는 함수입니다. 매개 변수로 x,y 좌표값을받아 커서를 이동시킵니다. Int형(0)을 반환하지만 실제로 사용되지는 않습니다.**

|  |  |
| --- | --- |
| int 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 0;  } | [cs](http://colorscripter.com/info#e) |

**2.SetColor함수: 색을 표현하는 함수입니다. 매개 변수로 색값을 가져와 색을 설정합니다. 반환값은 없습니다.**

|  |  |
| --- | --- |
| //출력글씨 색 설정  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    } | [cs](http://colorscripter.com/info#e) |

**3.Init함수: 게임 시작 시 변수를 초기화 하는 함수입니다. Isgameover을 레퍼런스로 받고 다양한 경우에서 수정합니다. 게임에서 다양한 변수들을 초기화 시켜 줍니다.Int형(0)을 반환하지만 실제로 사용되지는 않습니다.**

|  |  |
| --- | --- |
| int init(int& isgameover)  {      int i, j;//반복문 i,j 변수 선언        srand((unsigned)time(NULL)); //시드 값 시간으로  if(isgameover)  score=0;      isgameover = 0;        for (i = 0; i < 20; i++)//y좌표      {          for (j = 0; j < 14; j++) //x좌표          {              if ((j == 0) || (j == 13)) //x좌표가 0또는 13일때 - 왼쪽 오른쪽 벽 블록 설정              {                  total\_block[i][j] = 1; //블록              }              else {                  total\_block[i][j] = 0;   //아니면 빈블록              }          }      }        for (j = 0; j < 14; j++)         //화면의 제일 밑의 줄은 1로 채운다. 맨밑줄          total\_block[20][j] = 1;        //전역변수 초기화      level = 0; //지금 레벨 초기화      lines = 0; //지금까지 깬 라인 초기화      ab\_x = 5; //블록이 처음 시작하는 절대 좌표      ab\_y = 1;        //스테이지 별 스피드, 스틱 비율, 깨야하는 라인 수 초기화      stage\_data[0].speed = 40;      stage\_data[0].stick\_rate = 20;      stage\_data[0].clear\_line = 20;      stage\_data[1].speed = 38;      stage\_data[1].stick\_rate = 18;      stage\_data[1].clear\_line = 20;      stage\_data[2].speed = 35;      stage\_data[2].stick\_rate = 18;      stage\_data[2].clear\_line = 20;      stage\_data[3].speed = 30;      stage\_data[3].stick\_rate = 17;      stage\_data[3].clear\_line = 20;      stage\_data[4].speed = 25;      stage\_data[4].stick\_rate = 16;      stage\_data[4].clear\_line = 20;      stage\_data[5].speed = 20;      stage\_data[5].stick\_rate = 14;      stage\_data[5].clear\_line = 20;      stage\_data[6].speed = 15;      stage\_data[6].stick\_rate = 14;      stage\_data[6].clear\_line = 20;      stage\_data[7].speed = 10;      stage\_data[7].stick\_rate = 13;      stage\_data[7].clear\_line = 20;      stage\_data[8].speed = 6;      stage\_data[8].stick\_rate = 12;      stage\_data[8].clear\_line = 20;      stage\_data[9].speed = 4;      stage\_data[9].stick\_rate = 11;      stage\_data[9].clear\_line = 99999;      return 0;  } | [cs](http://colorscripter.com/info#e) |

**4.Show\_cur\_block함수: 블럭의 모양, 각도, x, y 좌표정보들을 매개변수로 받아 블록을 출력하는 함수입니다. Int형(0)값을 반환하지만 셀제로 사용되지는 않습니다.**

|  |  |
| --- | --- |
| int show\_cur\_block(int shape, int angle, int x, int y)  {      int i, j;      //정의 해둔 SetColor함수를 이용하여 모양에 따른 색 출력 구현      switch (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 0;      }      for (i = 0; i < 4; i++)      {          for (j = 0; j < 4; j++)          {              if ((j + y) < 0)// 출력되는 y좌표가 0이하 콘솔 창밖을 벗어날때는 출력하지 않음                  continue;                if (block[shape][angle][j][i] == 1) //해당하는 도형의 j,i위치가 빈블록이 아닐 때              {                  gotoxy((i + x) \* 2 + ab\_x, j + 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 0;  } | [cs](http://colorscripter.com/info#e) |

**5.erase\_cur\_block함수: 블록의 모양, 각도, x,y 좌표정보들을 매개로 받아 출력 되었었던 블록을 빈 칸으로 그래돌 덮어 씌워 지워주는 함수입니다. Int형(0)을 반환하지만 실제로 사용되지는 않습니다.**

|  |  |
| --- | --- |
| int erase\_cur\_block(int shape, int angle, int x, int y)  {      int i, j;      if (shape <= 6 && shape >= 0)          for (i = 0; i < 4; i++)          {              for (j = 0; j < 4; j++)              {                  if (block[shape][angle][j][i] == 1)//해당하는 블록위치가 빈 블록이 아닐 때                  {                      gotoxy((i + x) \* 2 + ab\_x, j + y + ab\_y);//이 위치에 빈블록으로 지움                      printf("  ");                  }              }          }      return 0;  } | [cs](http://colorscripter.com/info#e) |

**6.Show\_total\_block함수: 전체 블록(움직이고 있는 블록제외)을 출력해주는 함수입니다. 매개로 받는 변수는 없습니다.**

|  |  |
| --- | --- |
| int show\_total\_block()  {      int i, j;      SetColor(DARK\_GRAY);// 전체 블록, 색 설정 다크그레이      for (i = 0; i < 21; i++)      {          for (j = 0; j < 14; j++)          {              if (j == 0 || j == 13 || i == 20)      //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 0;  } | [cs](http://colorscripter.com/info#e) |

**7.Show\_next\_block함수: 게임진행시 생성되는 다음 블록을 출력하는 함수입니다. 모양정보를 매개변수로 받고 Int형(0)을 반환하지만 실제로 사용되지는 않습니다.**

|  |  |
| --- | --- |
| int show\_next\_block(int shape)  {      int i, j;      SetColor((level + 1) % 6 + 1); //다음블록이 표기되는 테두리는 레벨따라 색이 다름      for (i = 1; i < 7; i++) //다음에 무슨 블록이 나올지 알려주는 네모난 공간을 만듦      {          gotoxy(33, i);          for (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("  ");              }            }      }      show\_cur\_block(shape, 0, 15, 1);// 15\*2+ab\_x하면 35,  다음블록 안에다가 도형 출력      return 0;  } | [cs](http://colorscripter.com/info#e) |

**8.make\_new\_block함수: 새로운 블럭을 생성하는 함수 입니다. 여러조건에서 모양정보를 반환합니다. Int형값으로 0~6의 다양한 모양정보값들을 반환합니다.**

|  |  |
| --- | --- |
| int make\_new\_block()  {      int shape;      int i;      i = rand() % 100;      if (i <= stage\_data[level].stick\_rate)      //막대기 나올확률 계산          return 0;                     //막대기 모양으로 리턴        shape = (rand() % 6) + 1;      //shape에는 1~6의 값이 들어감      show\_next\_block(shape);      return shape;  } | [cs](http://colorscripter.com/info#e) |

**9.strike\_check함수: 게임중 블럭들의 충돌을 체크하는 함수입니다. 매개변수로 모양, 각도, x, y 좌표정보들을 받습니다. 조건값에 따라 Int형0~1을 반환합니다.**

|  |  |
| --- | --- |
| int strike\_check(int shape, int angle, int x, int y)  {      int i, j;      int block\_dat;        if (shape <= 6 && shape >= 0) {          for (i = 0; i < 4; i++)          {              for (j = 0; j < 4; j++)              {                  if ((x + j) == 0 || (x + j) == 13)                      block\_dat = 1;   //이 블록이 지금 토탈 블록의 외곽부 이면 -1                  else if (y + i < 0) //이경우 total\_block  에는 이부분이 없기에 추가  수정--                      block\_dat = 0;                  else                      block\_dat = total\_block[y + i][x + j];  //이블록이 토탈블록에 빈블록인지,찬 블록인지                      if ((block\_dat == 1) && (block[shape][angle][i][j] == 1))   //좌측벽의 좌표를 빼기위함? -- 이블록이 빈블록이 아니고 도형의 이 블록 부분도 1이면 충돌 !                  {                      return 1;                  }              }          }          return 0;      }      return 1;  } | [cs](http://colorscripter.com/info#e) |

int strike\_check(int shape, int angle, int x, int y); //블럭이 화면 맨 아래에 부닥쳤는지 검사 부닥치면 1을리턴 아니면 0리턴

**10.merge\_block함수: 게임중 블럭에 대해서 현재 블록과 토탈 블록을 병합하는 함수입니다. 매개변수로는 블록의 모양,각도,좌표값을 받아 전체블록과 병합하고 출력합니다. Int형(0)반환하지만 실제로 사용되지는 않습니다.**

|  |  |
| --- | --- |
| int merge\_block(int shape, int angle, int x, int y)  {      int i, j;      for (i = 0; i < 4; i++)      {          for (j = 0; j < 4; j++)          {              total\_block[y + i][x + j] |= block[shape][angle][i][j]; //토탈 블록에 현재블록 부분을 1로 바꿔줌          }      }      check\_full\_line(); //정의된 check\_full\_line함수  토탈블록이 변화 했으니 줄이 완성 되었으면 지워주기 위해      show\_total\_block();//전체 블록 출력        return 0;  } | [cs](http://colorscripter.com/info#e) |

**11.block\_start함수: block의 초기위치를 지정하는 함수**

|  |  |
| --- | --- |
| int block\_start(int shape, int\* angle, int\* x, int\* y)  {      \*x = 5;      \*y = -3;      \*angle = 0;      return 0;  } | [cs](http://colorscripter.com/info#e) |

int block\_start(int shape, int\* angle, int\* x, int\* y); //블럭이 처음 나올때 위치와 모양을 알려줌

**12.move\_block함수:** 함수 설명 : 블럭을 1칸 내리는 함수이다. Return 값은 현재 블럭 상태를 의미하는데, 0은 충돌하지 않은 상황, 1은 게임오버로 충돌하는 상황, 2는 더 이상 내려갈 수 없는 상황을 나타낸다.

|  |  |
| --- | --- |
| int move\_block(int\* shape, int\* angle, int\* x, int\* y, int\* next\_shape)  {      erase\_cur\_block(\*shape, \*angle, \*x, \*y);//지금 현재의 블록을 지워주고        (\*y)++;   //블럭을 한칸 아래로 내림      if (strike\_check(\*shape, \*angle, \*x, \*y) == 1) //한칸을 내렸을 때 충돌했으면      {          if ((\*y) < 0)   //게임오버          {              total\_block, x, y, shape; //테스트 구문 --제출시 삭제              return 1; //충돌하였다 입력          }          (\*y)--; //충돌한 것이 화면 밖에서 일어난 것이 아니면 다시 이전위치로 복구,          merge\_block(\*shape, \*angle, \*x, \*y); //맨 끝까지 내려온것이므로  토탈 블록과 블록을 병합          (\*shape) = \*next\_shape;            //모양 다음 블록에서 가져옴          (\*next\_shape) = make\_new\_block(); //다음 블록 재생성            block\_start(\*shape, angle, x, y);   //angle,x,y는 포인터임          show\_next\_block(\*next\_shape);   //다음블록 출력함수 오른쪽 위에          return 2; //내려 올수 있는 곳까지 내려왔음을 리턴      }        //한칸 내렸을 때 충돌 안햇으면      return 0;  } | [cs](http://colorscripter.com/info#e) |

**13.rotate\_block함수: - 프로그램에서 사용하지 않아 분석 생략**

**14.Show\_gameover함수:** gameover 화면을 출력함과 동시에 쌓여있을 입력버퍼를 지워 향후 지장이 없도록 하는 함수이다.

Return 값은 사용되지 않는다.

|  |  |
| --- | --- |
| int show\_gameover()  {      //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 0;  } | [cs](http://colorscripter.com/info#e) |

**15.Show\_gamestat함수: 게임에서 점수판을 출력하는 함수이다. 받은 매개변수는 없고, 스테이지레벨, 점수, 라인에 대한 정보를 화면에 출력한다. Int(0)을 반환하지만 실제로 사용되지는 않는다.**

|  |  |
| --- | --- |
| int 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?      static int printed\_text = 0; // 밑에있는 조건문의 조건에 들어가는 변수 조건문을 실행해줌        SetColor(GRAY);      if (printed\_text == 0)      {          gotoxy(35, 7);          printf("STAGE");            gotoxy(35, 9);          printf("SCORE");            gotoxy(35, 12);          printf("LINES");          }      gotoxy(41, 7);      printf("%d", level + 1);      gotoxy(35, 10);      printf("%10d", score);      gotoxy(35, 13);      printf("%10d", stage\_data[level].clear\_line - lines);      return 0;  } | [cs](http://colorscripter.com/info#e) |

**16.Show\_logo함수: 게임에서 점수판을 출력하는 함수이다. 받은 매개변수는 없고, 키를 입력받으면 함수가 종료된다. Int형(0)을 출력하지만 사용되지는 않는다.**

|  |  |
| --- | --- |
| int show\_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 (j = 0; j < 5; j++)              {                  gotoxy(18, 14 + j);                  printf("                                                          ");                  }              for (int i = 14; i < 19; i++) {                  gotoxy(6, i); printf("%25s", ""); //수정-잔상지우기위해              }              show\_cur\_block(rand() % 7, rand() % 4, 6, 14);              show\_cur\_block(rand() % 7, rand() % 4, 12, 14);              show\_cur\_block(rand() % 7, rand() % 4, 19, 14);              show\_cur\_block(rand() % 7, rand() % 4, 24, 14);          }          if (kbhit())              break;          Sleep(30);      }        getche();      system("cls");        return 0;  } | [cs](http://colorscripter.com/info#e) |

**17.input\_data함수: 게임세작시 도움말을 출력하고 스테이지를 입력받는 함수이다. 매개로 받는 변수는 없다. 게임키에 대한 도움말을 출력하고 스테이지 레벨을 int형으로 1보다 크고 8보다 작은수를 입력받는다. 입력받은수는 변수 level에 반영된다. Int형(0)을 반환하지만 실제로 사용되지는 않는다.**

|  |  |
| --- | --- |
| int 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);          }          \*/        }          level = i - 1;      system("cls");      return 0;  } | [cs](http://colorscripter.com/info#e) |

**18.check\_full\_line함수: 게임중, 블록을 가지고 줄 전체를 가득 채우면, 해당줄을 제거하고 토탈 블록 데이터를 수정하는 함수이다. 매개로 받는 변수는 없다. Int형(0)을 반환하지만 사용되지는 않는다.**

|  |  |
| --- | --- |
| int check\_full\_line()// 리턴이 int로 되있긴 하지만 리턴값을 사용하지 않음  {      int i, j, k;//i-y좌표, j-x좌표 ,k      for (i = 0; i < 20; i++) //전체 모든 줄 체크      {          for (j = 1; j < 13; 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          {              lines++; //지금까지 깬 라인 +1                //출력-동작              show\_total\_block(); //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 (k = i; k > 0; k--)//지금 줄 i부터 그 윗줄 반복              {                  for (j = 1; j < 13; j++)                      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;              score += 100 + (level \* 10) + (rand() % 10); //점수 레벨에따른 상승              show\_gamestat(); //게임 정수판 수정          }      }      return 0;  } | [cs](http://colorscripter.com/info#e) |

**19.main함수: 주 실행함수**

|  |  |
| --- | --- |
| int main(int argc, char\* argv[])//메인의 인자 값은 사용하지 않는 듯  {      int i;      int is\_gameover = 0; //처음 is\_gameover 0으로 초기화      char keytemp;      init(is\_gameover);      show\_logo();      while (1)      {          input\_data(); // 게임 스테이지 설정          show\_total\_block();   //토탈 블록 출력          block\_shape = make\_new\_block(); //처음 시작 할 때-움질일블록 새로운 블록 생성          next\_block\_shape = make\_new\_block(); // 다음 블록 -새로운 블록생성          show\_next\_block(next\_block\_shape); //다음 블록 출력          block\_start(block\_shape, &block\_angle, &block\_x, &block\_y);//블록 시작 초기값함수          show\_gamestat();          for (i = 1; 1; i++)          {              if (kbhit())//키 입력이 들어온다면              {                  keytemp = getche();                  if (keytemp == KEY\_ESC) {        //esc입력이 들어오면                      is\_gameover = 1; //게임오버 설정                      system("cls");                      break;                  }                  if (keytemp == EXT\_KEY) //확장키 인식값                  {                      //방향키 등의 특수키는-확장키+특수키분류의 두번이 입력됨                      keytemp = getche();                      switch (keytemp)                      {                      case KEY\_UP:      //회전하기-방향위키 입력시                            if (strike\_check(block\_shape, (block\_angle + 1) % 4, block\_x, block\_y) == 0)                          {//각도를 한번 돌렸을 때 충돌하지 않으면, 현재 블록 출력 지우고,각도 바꿔주고, 다시 출력                              erase\_cur\_block(block\_shape, block\_angle, block\_x, block\_y);                              block\_angle = (block\_angle + 1) % 4;                              show\_cur\_block(block\_shape, block\_angle, block\_x, block\_y);                          }                          break;                      case KEY\_LEFT:      //왼쪽으로 이동-                          if (block\_x > 1)//현재 블록의 위치가 맨 왼쪽이 아니면(블록 모양상관없이)                          { //지금 블록의 출력부를 지우고                              erase\_cur\_block(block\_shape, block\_angle, block\_x, block\_y);                              block\_x--; //한칸 왼쪽으로 이동하고                              if (strike\_check(block\_shape, block\_angle, block\_x, block\_y) == 1)                                  block\_x++; //충돌시에는 다시 복귀                              //현재 블록 출력                              show\_cur\_block(block\_shape, block\_angle, block\_x, block\_y);                          }                          break;                      case KEY\_RIGHT:      //오른쪽으로 이동                            if (block\_x < 14)                          { //현재 블록의 위치가 맨 오른쪽이 아니면, 지금 블록의 출력부를 지우고                              erase\_cur\_block(block\_shape, block\_angle, block\_x, block\_y);                              block\_x++; //한칸 오른쪽으로 이동                              if (strike\_check(block\_shape, block\_angle, block\_x, block\_y) == 1)                                  block\_x--; //충돌시에는 다시 복귀                              //현재 블록 출력                              show\_cur\_block(block\_shape, block\_angle, block\_x, block\_y);                          }                          break;                      case KEY\_DOWN:      //아래로 이동                          //아래로 이동 moveblock에서 1이 리턴 게임 오버                          is\_gameover = move\_block(&block\_shape, &block\_angle, &block\_x, &block\_y, &next\_block\_shape);                          if(is\_gameover != 1) //게임 오버 되지 않았을 때에만 -게임 오버 되었을때는 다음블록이 바뀌지 않으므로                              show\_cur\_block(block\_shape, block\_angle, block\_x, block\_y);                          break;                      }                  }                  if (keytemp == 32)   //스페이스바를 눌렀을때                  {                      while (is\_gameover == 0)                      {    //맨 바닥에 닿으면 move\_block가 2를 출력하는데 스페이스를 누르면 내려갈수 있는 곳까지 내려가기 위해                          is\_gameover = move\_block(&block\_shape, &block\_angle, &block\_x, &block\_y, &next\_block\_shape);                      }                      if (is\_gameover != 1) //is\_gameover가 1이 아닐 때에만 다음 블록 출력                          show\_cur\_block(block\_shape, block\_angle, block\_x, block\_y);                  }              }              if (is\_gameover == 1)//게임오버 상태라면              {                  show\_gameover();//게임오버 로고 출력                  SetColor(GRAY); //색 설정                  break;              }              if (i % stage\_data[level].speed == 0)//i가 상승하다가~i가 스테이지의 스피드 배수 마다  한칸씩 내려가게              {    // 한칸 내려가는 move\_block 함수 실행  리턴값 is\_gameover                  is\_gameover = move\_block(&block\_shape, &block\_angle, &block\_x, &block\_y, &next\_block\_shape);                  //블록 출력                  show\_cur\_block(block\_shape, block\_angle, block\_x, block\_y);              }              //만약 깬 라인이  스테이지의 클리어 라인 이상이면 다음 스테이지로 넘어가기              if (stage\_data[level].clear\_line <= lines)   //클리어 스테이지 라인 1개 남았을  때 2개이상의 라인을 깼을 경우 클리어 되지 않아 <추가              {                  level++; //레벨+1                  lines = 0; //깬 라인 0으로                  show\_gamestat(); //스텟초기화              }                  gotoxy(77, 23);              Sleep(15);              gotoxy(77, 23);          }          init(is\_gameover); //게임 재시작 마다 변수 초기화 함수 실행      }      return 0;  } | [cs](http://colorscripter.com/info#e) |

**-실행결과:**

**<게임 시작 로고 수정>**



**-> 게임 시작 시, 화면이 깨지는 것을 수정하였다.**

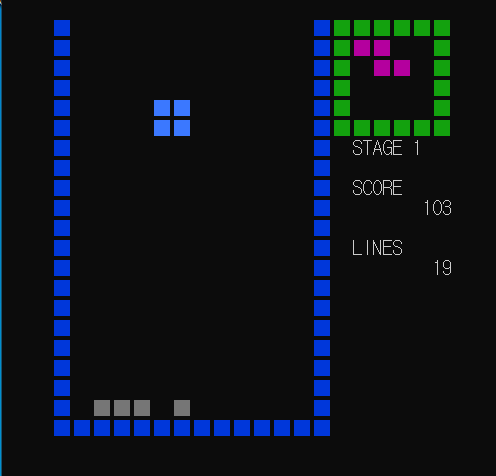
**<문자 입력 오류>**





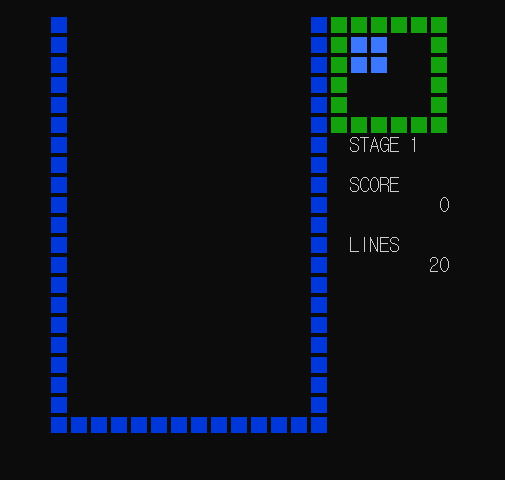
**-> 원래 정수가 아니라, 문자를 받으면 무한루프가 발생하는 현상이 있었으나, 이를 해결하였다.**

**<한 줄 지우기>**



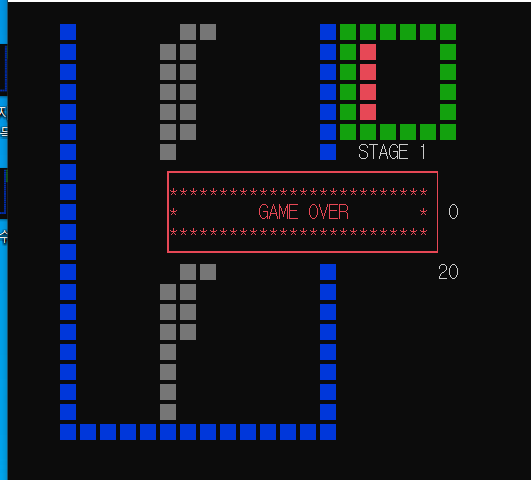
**-> Lines가 하나 줄고 정상적으로 점수가 올라갔다.**

**<esc로 뒤로 가기>**



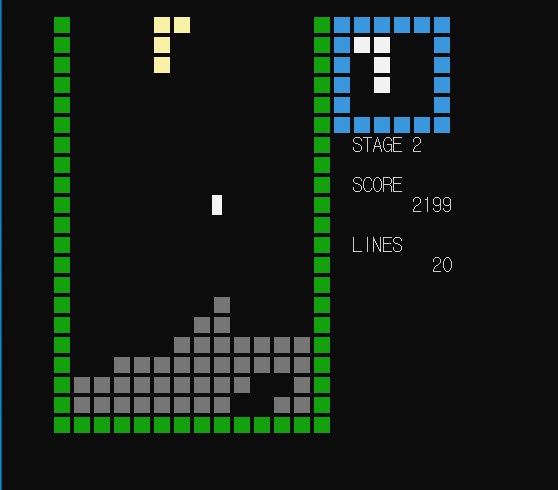
**-> 기존의 esc를 누르면 잔상이 남는 버그가 있었는데, 잔상을 제거하는 대신 뒤로 가는 기능으로 바꾸었다. 점수 또한, 초기화되는 모습을 볼 수 있다.**

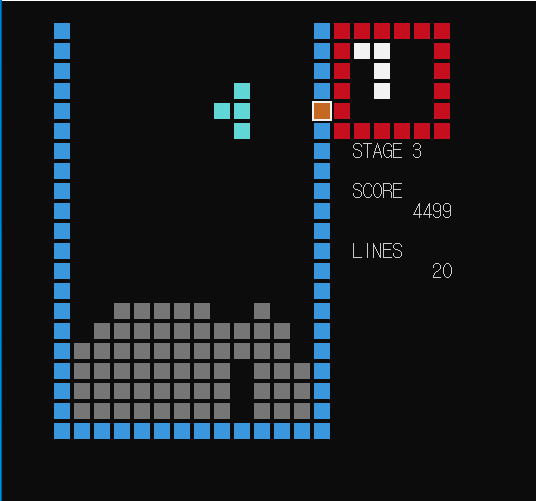
**<게임 오버 및 로고 수정>**



**-> 게임 오버가 되었을 경우, 게임 오버 로고 창을 띄우고 점수를 초기화한다. 이때, 게임 오버 로고 창도 깨지는 버그가 있어서 수정하였다.**

**<스테이지 클리어>**





**-> 스테이지를 클리어하면, 외벽 색이 바뀌면서 LINES가 초기화된다.**

**그리고, LINES = 1에서 한번에 2줄을 지워서 -1이 되도 자동으로 스테이지가 클리어 되도록 조정하였다.**

**느낀 점 및 토의 사항:**

* **게임을 실행해 보면서 변수, 함수들이 어떤 기능을 하는지 알아보고 버그를 찾아 고쳐보았다. 게임을 실행할 때 생기는 버그들이 왜생기는지 코드를 보며 생각해 보았고 이는 프로그램을 짤 때 고려해야 할 것들이 많고 테스트도 잘해보아야함을 느꼈다. 좀더 생각해 보아야할 문제로는 버그를 찾았는데 그 해결 방안에 있다. 어떤 버그를 어떻게 고쳐야 좋을지 조금더 생각해 보아야 겠다.**
* **예상외로 발견된 버그가 많았다. 아무래도, 처음 보는 코드를 쭉 읽어보면서 찾다보니까 버그도 잘 찾아지지 않았고, 해결하는 데에도 시간이 오래 걸렸다. 이러한 코드를 보면서, 내가 코드를 짤 때에는 미리 조금씩 테스트하여 오류를 줄이는 습관을 길러야겠다고 생각하였다.**
* **평소 버그를 테스트하고 수정하는 건 한 단계씩 진행하곤 했는데 거의 완성된 프로그램을 수정하는 건 처음이었다. 나중에 관련된 일을 하게 되면 분명 겪을 상황이기에 좋은 경험이 되었다고 생각한다. 또한 평소와 다르게 새로운 문제를 해결하는 것이 아닌 버그를 수정하기 위해 알고리즘을 생각하는 것 역시 색다른 경험이었다고 생각한다. 앞으로도 버그를 해결할 때에 있어서 이번 경험은 아주 큰 도움이 될 것이라고 생각한다.**