**C++ 프로그래밍**

**설계 프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | 테트리스 게임 만들기 |
| 소속 | *국민대학교 경제학과* |
| 학번 | *2013094040* |
| 이름 | *장용훈* |
| 문서 제목 | mAKE TETRIS |
| 지도 교수 | 최진우 |

|  |  |
| --- | --- |
| **Version** | 30 |
| **Date** | 2017-05-30 |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | MAKE TETRIS |
| **원안작성자** | 장용훈 |
| **수정작업자** | 장용훈 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2017-04-09 | 장용훈 | 1.0 | 기획 | 초안 작성 |
| 2017-04-10 | 장용훈 | 1.1 | 수정 | 함수설계 추가 |
| 2017-05-17 | 장용훈 | 1.2 | 수정 | 클래스 작성 |
| 2017-05-30 | 장용훈 | 1.3 | 최종 | 구현 완료 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**목 차**

[1 개요 4](#_Toc347414841)

[1.1 프로젝트 개요 4](#_Toc347414842)

[1.2 추진 배경 및 필요성 4](#_Toc347414843)

[2 개발 내용 및 결과물 5](#_Toc347414844)

[2.1 목표 5](#_Toc347414845)

[2.2 연구/개발 내용 및 결과물 6](#_Toc347414846)

[2.2.1 연구/개발 내용 6](#_Toc347414847)

[2.2.2 활용/개발된 기술 6](#_Toc347414849)

[2.2.3 현실적 제한 요소 및 그 해결 방안 6](#_Toc347414850)

[2.2.4 결과물 목록 6](#_Toc347414851)

[2.3 기대효과 및 활용방안 6](#_Toc347414852)

[3 자기평가 7](#_Toc347414853)

[4 참고 문헌 7](#_Toc347414854)

# 개요

## 프로젝트 개요

**텍스트 사용자 인터페이스(Text User Interface, TUI) 라이브러리 중 하나인 NCurses를 활용하여 콘솔창에서 작동하는 테트리스 게임을 디자인 하도록 한다**

## 추진 배경 및 필요성

**유닉스 계열 운영체제인 Curses를 학습하여 텍스트 사용자 인터페이스(TUI) 응용프로그램들의 구성을 가능케 하고 윈도우, DOS등을 위한 공개 라이브러리 NCurses 를 이용해 프로그램을 작성한다.**

# 개발 내용 및 결과물

## 목표

1. 테트리스의 기능을 어떻게 구현 할 것인가에 대하여 생각해본다.

- 테트리스의 규칙, 그리고 블록의 모양에 관하여 조사한다.

1. 개발환경을 어떻게 구성할지에 대하여 생각해본다.

- Vi에 NCurses 라이브러리를 적용하여 코드를 작성한다.

1. UI관점에서 코드를 구현 하기 전 밑 그림을 그려본다.

- 각종 자료들을 참고하여 미리 설계도를 그려본다.

4. 클래스의 관점으로 함수를 미리 정의 해본다,

5. 설계도를 바탕으로 구현을 한다.

6. 각 함수의 알고리즘을 정의한다.

7. 각 블록의 상태를 정의한다.

8. 블록의 회전 알고리즘을 정의한다.

9. 상속의 개념을 이용하여 블록 클래스를 설계한다.

10. 키보드 입력모드와 파일 입력모드를 구현한다.

11. 초기 창에서 이름을 입력 받아, 이를 출력을 해준다.

## 연구/개발 내용 및 결과물

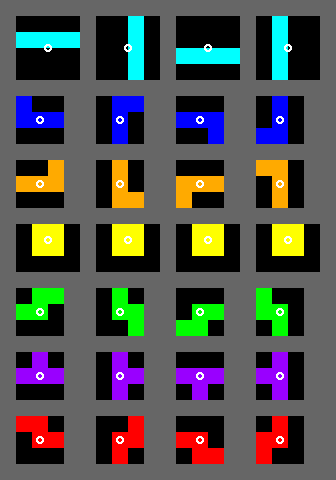
### 연구/개발 내용

* 테트리의 정의

1. **네 개의 사각형으로 이루어진 '테트로미노'는 무작위로 나타나 바닥과 블록 위에 떨어진다. 이 오락의 목표는 이 '테트로미노'를 움직이고 90도씩 회전하여, 수평선을 빈틈 없이 채우는 것이다. 이러한 수평선이 만들어질 때 이 선은 없어지며 그 위의 블록이 아래로 떨어지는데 테트리스 게임이 진행될수록 '테트로미노'는 더 빨리 떨어지며 게임을 즐기는 사람이 블록을 꼭대기까지 가득 메워, '테트로미노가 더 들어갈 공간이 없게 되면 게임이 끝나게 된다.**
2. **테트리스 게임의 매뉴얼은 테트리스에서 *I*, *J*, *L*, *O*, *S*, *T*, *Z*와 같이 7개의 단면 '테트로미노'를 일컫는다.**[**[12]**](https://ko.wikipedia.org/wiki/%ED%85%8C%ED%8A%B8%EB%A6%AC%EC%8A%A4#cite_note-tdsmanual-12)**비슷한 글자 모양을 가지고 있기 때문에 이렇게 일컫지만, 게임을 즐기는 사람들은 이러한 조각을 다르게 부르기도 한다.**[**[13]**](https://ko.wikipedia.org/wiki/%ED%85%8C%ED%8A%B8%EB%A6%AC%EC%8A%A4#cite_note-arkmay_pieces-13)

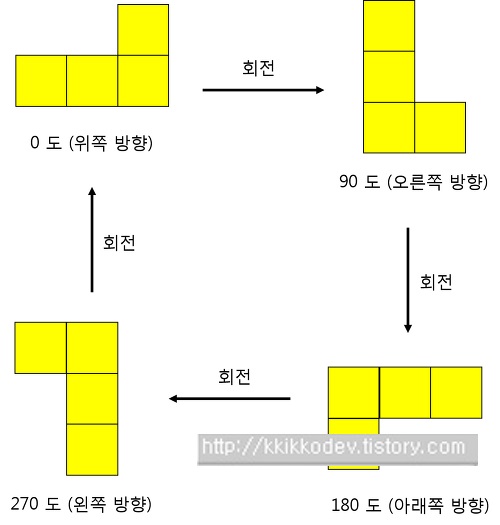
* **테트리스 블록의 정의**

**[블럭의 종류 7 가지 형태]**



**블럭의 종류는 위 그림과 같이 총 7개로 이루어져 있다. 하나의 블럭은 네모 상자 4 개가 붙여져 이루어지기 때문에, 각각 네모 상자 하나마다, x, y 좌표를 가지게 됩니다. 따라서, 블럭 1 개는, 4 개의 좌표를 관리해야 한다.**

**[블럭이 90 도씩 회전 하는 경우 모양]**

****

**그리고 이각각의 블록은 회전할 수 있음으로 이는 배열로 관리한다.,**

* 설계 일정

1단계

1개발환경 결정 및 초기화면 구성

1테트리스의 규칙 및 블록에 대한 정의

2단계

사용할 함수 및 변수에 대한정의

정의한 부분을 헤더파일화

각각의 함수 구현

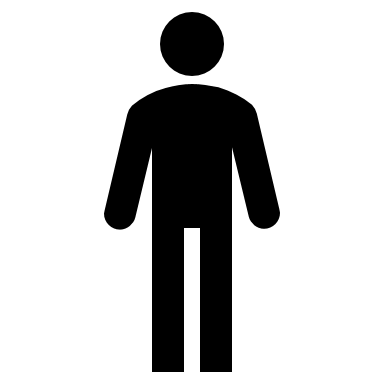
3단계

만들어진 함수들을 조정하여 실행

디버깅

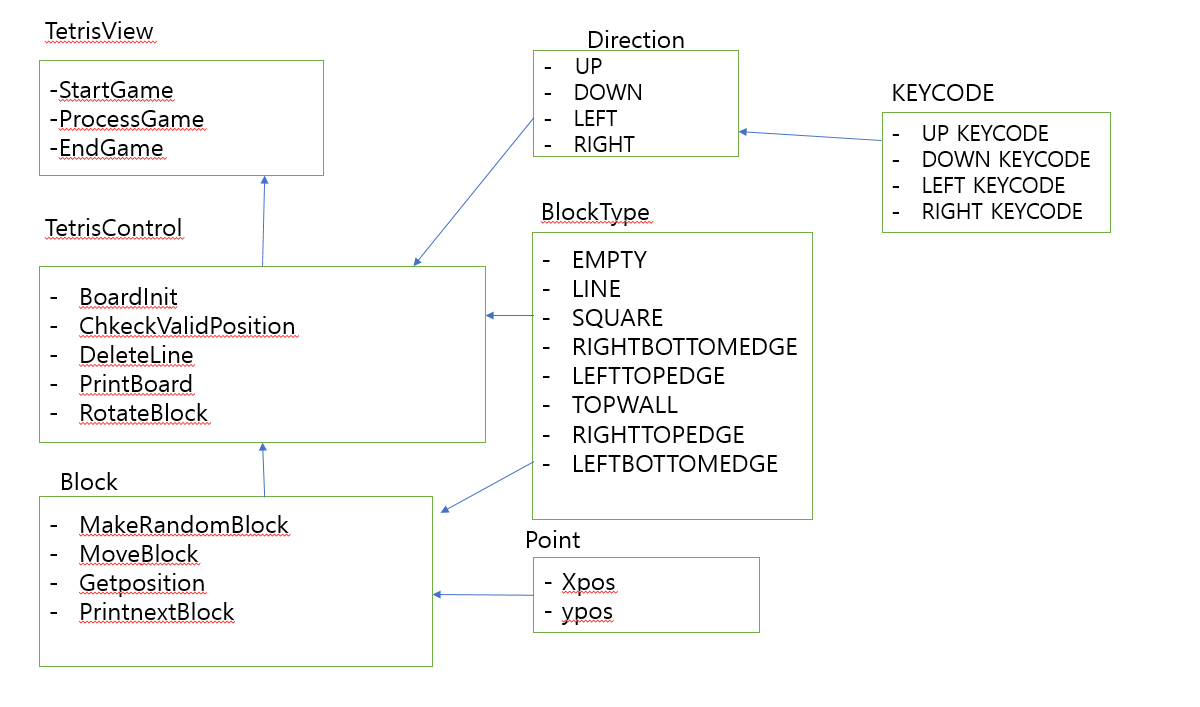
평가 및 보고서 작성

* UI관점에서 설계



사용자가 키를 입력하여 실행하거나 혹은 종료를 할 수 있게 선택을 할 수 있다. 만약 실행을 한다면 동시에 블록 밑 테트리스판이 윈도우 창안에 새로운 윈도우 창이 생성되어 나타난다. 이때 블록은 생성됨과 동시에 키조작을 통하여 블록을 이동시킬 수 있다. 이때 특정키를 눌러 종료시킬 수 있다. 혹은 게임오버 상황이 된다면 테트리스 블록은 더 이상 생성 되지 않고 종료된다.

* 기능 및 클래스 설계



1. TetrisView

- StartGame : 게임을 시작하는 함수이다. 호출시 메인 보드를 화면에 나타낸다.

- ProcessGame : TetrisConrol클래스의 함수를 호출하여, 상태의 변화를 계속해서 업데이트 한다.

- EndGame : 호출시 ProcessGame을 중단시키고, 화면에 END라는 단어가 표시된다.

2. TetrisControl

- BoardInit : 테트리스를 시작 할 수 있게 새로운 보드판을 띄어준다. 뿐만 아니라 새로운 보드를 생성 할 수도 있다.

- CheckValidPosition : DeleteLine이 블록을 지우기전, 지울 수 있는지 여부를 체크해 주는 함수이다.

- DeleteLine : CheckValidPosition을 호출하여 값을 리턴받아, 그 값에 따라 조건에 맞을 경우 줄의 라인을 지운다.

- PrintBoard : 각각의 함수실행의 값을 입력받아 콘솔에 업데이트를 해주는 역할을 하는데, 실제로 화면의 업데이트는 ProcessGame에서 이루어진다.

- RotateBlock : 블록의 회전을 담당하는 함수이다. 여기에 각각 블록 배열이 선언이 되어 있고 키의 입력에 따라 그 블록의 모양이 달라진다.

3. Block

- MakeRandomBlock : 블록을 생성하는데, 이때 난수를 이용하여 해당 하는 값의 블록을 만들어서 반환한다.

- MoveBlock : 상태창에 블록을 새로 생성된 블록으로 교체해주는 함수이다.

- GetPosition : x좌표와 y좌표의 값을 얻어주는 역할을 한다. 테트리스를 게임 작동을 위한 핵 심적인 함수 중 하나이다.

- PrintNextBlock : MoveBlock으로 인해 교체된 블록을 다른 윈도우 창에 띄어주는 역할을 한다.

4. Direction

- UP, DOWN, LEFT, RIGHT로 이루어져 있는데 각각이 그에 해당하는 KEYCODE 값을 받을 때, 블록이 이동 할 수 있게 해준다.

5. KEYKODE

- 화살표 방향과 스페이스를 키로 이용한다.

6. BlockType

- 각각의 선언된 값은 그에 해당하는 블록의 정보를 가지고 있다. 설계하고자 하는 테트리스에서 블록의 개수를 7개로 정의 하였기에, 이에 대응하는 블록의 개수를 가지고 있다. 그리고 여기에 블록이 비어 있는 경우를 추가하였다.

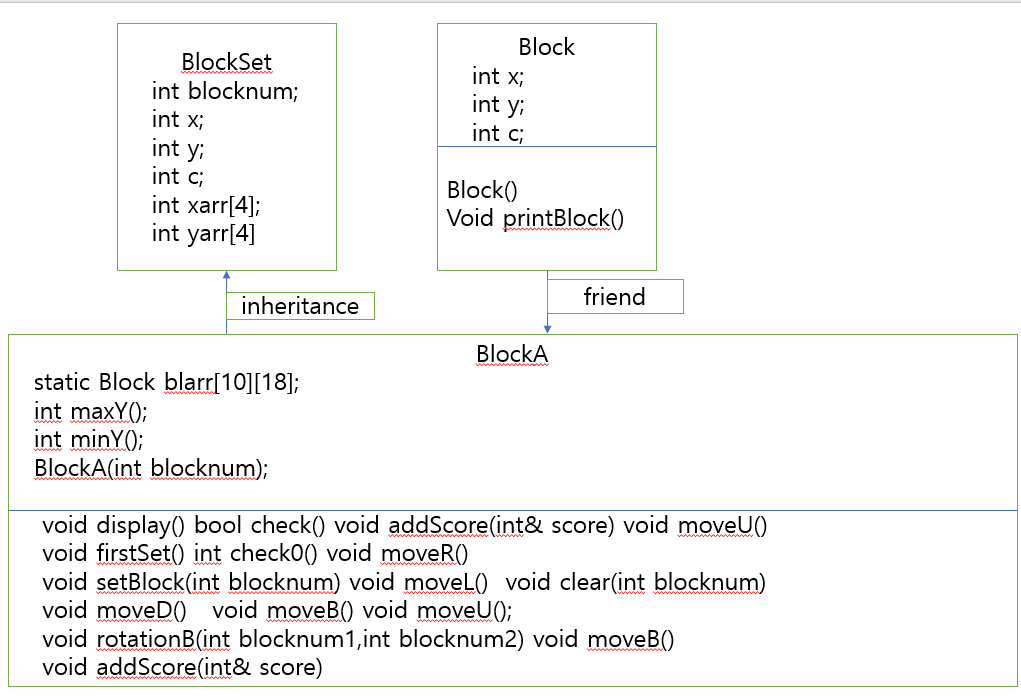
7. Point

- x와y좌표 한쌍으로 이루어져 있고, 이를 이용하여 활용하여 테트리스가 작동을 하게 된다.

이와 같이 총 7개의 커다란 틀로 이루어져 있고, 그 큰 틀안에 함수혹은 상수가 선언 되어있어, 유기적으로 이를 이용하게 된다.

2차과제에서의 활용은 다음과 같이 전개 되었다.

1. 기존의 설계해둔 방향을 완전히 뒤엎었다. 기존의 설계에서 불필요한 부분이 있을 뿐만 아니라, 기존의 조사와 새로운 자료조사의 결과가 일치하지 않았다. 뿐만 아니라 구현의 어려움도 발견 되었다.



따라서 다음과 같이 클래스 선언과, 그 활용 방법을 변경 하게 되었다.

그리고 이를 바탕으로 선언부와 구현부를 나누어 구현 하였다.

1. . 핵심적인 아이디어는 다음과 같다.

2차원 배열 선언을 다음과 같이 해준다.

10000001

10000001

10000001

10000001

10000001

11111111

이때,1이 경계값이 되고, 블록은 1을 넘어 갈 수 없다

각 부분의 좌표들은 시행착오를 통하여 구해야 한다.

라인 클리어는 다음과 같다.

해당줄이 0이 아닐 때 줄을 지워주면 된다. 다시 말해 배열에 블록이 쌓이는데 이때의 값을 일일이 비교를 해주고, 해당행의 모든 값 들이 0이아니면 줄을 지워준다. 이때 여러 줄을 지우고자 하면 이 행위를 반복하게 수행하게끔 만들 면 된다.

블록 생성은 배열에 블록을 넣어주고 그 인덱스 값을 랜덤하게 주면 된다.

각 블록은 블록클래스의 케이스별로 x,y좌표로 이루어진 2차원 배열의 초기화를 각각 다르게 해주면 된다. 2차과제에서는 사각형 블록만 출력되면 되기 때문이 이 블록만 설정 해 주면 된다.

그리고 출력은 블록의 2차원 배열을 넘겨받아서 각 행과 열의 값을 비교하여 조건을 만족하면 출력 조건을 만족하지 않다면 출력하지 않도록 하면 된다.

점수의 경우 클리어가 될 때 마다 그때 score점수를 더해주고, score를 나타내는 윈도우에 표시를 해주면, 조건을 만족 할 때마다 점수가 업데이트 된다.

3차 과제 이후의 블록의 모양 전개는 다음과 같다.

각 블록마다 좌표를 다르게 구해준다. 이 때 별도의 특별한 방법은 없고, 다만 시행착오를 통하여 블록의 좌표를 구해주어야 한다. 또한 이 블록들은 각각 회전을 할 수 있기에, 회전시켰을 때 모양의 좌표 또한 별도로 구해주어야 한다.

따라서 케이스마다 각각의 좌표가 만들어 지게 되고, 이를 선택하여 이용하기만 하면 된다.

### 활용/개발된 기술의 알고리즘

이번 1차설계에서는 다음과 같은 nCURSES의 함수가 이용되었다.

- 터미널 크기조정 resize\_term(a,b)

- 지정 위치에 문자열 출력 mvprintw(a,b,"블라블라");

- 터미널에 지금까지한 명령어가 적용되는 함수 refresh()

색사용 전에 반드시 선언 start\_color()

- 한쌍의 색 attribute지정 init\_pair(1,COLOR,COLOR)

- Attribute로 윈도우 전체 적용 bkgd(COLOR\_PAIR(1));

- 적용할 Attribute 설정 attron(COLOR\_PAIR(1))

- 적용한 Attribute 해제 attroff(COLOR\_PAIR(1))

- 윈도우 경계선 설정 border(ls,rs,ts,bs,tl,tr,bl,br) 사이드//코너

- 입력시 특수키 입력 가능 keypad(stdscr,TRUE)

- 화면에 보이는 커서 설정 curs\_set(0~2)

- 문자 입력시 입력한 값을 화면에 안보이게 noecho()

- 문자열 입력 scanw를 이용하였다.

- 윈도우에 문자열 출력 printw()

- 모두 지우기 clear()

- 윈도우를 나타내는 포인터 WINDOW\*var

- 새로운 윈도우 생성 win1=newwin(20,20,3,3)

- wbkgd wattron wborder wrefresh w만 빼면 위와 기능같음

- 할당한 윈도우 메모리 해제 delwin(w1);

- 이번 2차과제에서 활용된 알고리즘은 다음과 같다.

- 상속 : 우선 기본적인 정보를 담고 있는 blockset이라는 클래스를 만든다. 이를 모든 block에 대한 정보를 가지고 있는 blockA가 상속을한다. BlockA에는 blockset의 변수를 이용하여 이를 활용 하는 함수가 선언 되어 있다.

- Friend : block 클래스에는 printblock이라는 함수가 있는데, 콘솔창에 블록이 나타나려면 필수적으로 활용해야 하는 함수이다. 그렇기 때문에 block 에서 blockA에 대한 firend 선언을 하여 기능 적인 성격이 모여있는 곳에서 별다른 선언없이 이를 이용 할 수 있게 하였다.

- Block의 정의 : 2차원 배열을 이용 하여 x좌표 y좌표를 설정해주었다. 이때 값이 0이 아닐경우 색을 칠해주어, 결과적으로 이러한 모양 ■4개가 한 개의 묶음으로 나타내어 프린트 해준다.

- printBlock() : 이 함수는 특정 값을 받아 □ ■ 값에따라 block의 색깔을 다르게 해주어 결과적으로는 값만 있는 각 모양의 block들이 실질적으로 창에 나타나게 해준다.

- Blockset() : blockarr에 값을 초기화 해준다. 이 값들은 윈도우창에 줄을 나타내어 영역을 표시해 준다. 이때 2차원 배열을 사용해주어 경계과 되는 부분과, 안쪽이 되는부분(block이 채워질부분)을 구분을 하게 된다.

- MaxY, MinY : y좌표의 최대값 최소값을 리턴해준다.

- setBlock: 이번 과제에서는 사각형 블록만 있으면 되기 때문에, 사각형 블록에 대한 정보를 set에 주는 함수이다.

- Check : 정해진 좌표를 넘는 지 안넘는지 확인을 해주는 함수이다. 기존에 미리 구해논 창에 대한 바운더리를 넘지 못하게 해준다. 만약 값이 넘아간다면 false를 리턴해준다.

-:moveL,moveR : 호출시마다 블록의 X좌표 변화를 주어 결과적으로는 블록을 조정 할 수 있게 만들어 준다. 이 때 호출시마다 반드시 check를 사용하여 범위를 넘는지 넘지 않는지 확인을 해주어야 한다.

- moveD : 호출시마다 블록의 y좌표 변화를 주어 결과적으로는 블록을 조정 할 수 있게 만들어 준다. 이 때 호출시마다 반드시 check를 사용하여 범위를 넘는지 넘지 않는지 확인을 해주어야 한다.

- moveB : moveD와 유사한 기능을 한다. 다른점이 있다면 moveD의 경우에는 한칸만 동작을 한다면, moveB는 호출시 블록을 바닥까지 내려준다는 점이 가장 큰 다른점이다.

- addScore : 해당 줄의 값을 비교하여 조건을 만족하면 점수를 추가해 준다. 그 후 줄을 1칸 혹은 여러칸 땡겨준다. 줄을 지워 줄 때, 루프를 돌면서 일정한 조건을 만족 시 값을 계속해서 카운팅 해준다, 그후 반복문 밖에서 조건을 만족하면, 해당 열의 색을 바꿔준다.

- Init : curse모드를 시작해주고, 색깔을 설정해준다. 그리고 main window의 border를 설정해준다.

- printBackground : 각 윈도우 창의 사이즈를 설정해주고, wvorder를 나타내준다.

- display() : blockA에 키보드 값을 받은 후, 이 변화를 출력해준다.

Block이 일정시간 지나면, 떨어지게 하기위해 sleep()함수를 이용하였다. {}묶어, 일정 시간이 지나면 진행상황이 진전되도록 만들었다.

키 값의 구현은 다음과 같이 하였다. Q가 입력되면 종료 그리고 스페이스바가 입력되면 새로운 블록이 생성되고, 방향키에 따라서, 블록의 모양이 달라진다.

이번 3차과제에서 사용된 알고리즘은 다음과 같다.

- 가장 큰 다른 점은 블록이 기존의 ㅁ뿐만아니라 ㄱ ㄴㅣZ 등 총 5개의 블록을 추가 하였다. 이때 각 블록의 클래스를 만드는 것이 아닌 CASE로 관리 하였다. 0,2,6,10,11,13,15 각각의 숫자는 각각 다른 블록의 모양을 보여줄 뿐만 아니라, 이와 연동하여 각 블록의 색깔을 다르게 하였다. 이때도 역시 블록의 모양을 나타내 주기 위해 xarr,yarr 두 배열 좌표의 값을 각각 다르게 초기화 해주어 블록의 모양을 나타냈다.

블록을 랜덤으로 보여주기 위해 rand 함수를 이용했다. 그리고 완전한 랜덤번호를 얻기 위해 여기에 seed를 추가하여 실시간으로 블록이 떨어 진다면, 다른 블록이 혹은 같은 블록이 나올 수 있게 하였다.

- Rotation(num1, num2) : 블록의 회전을 담당 해주는 함수이다. 여기에 위에서 정의한 setblock 그리고 check를 이용하였다. Rotation이 호출 되면 check로 블록과 벽의 관계를 확인후 조건을 만족하면 setblock 호출하여 num2에 해당하는 블록을 출력 하게 해주었다. 이때 setblock.에는 기본적은 블록 뿐만 아니라, 블록을 회전 했을 때의 모습 까지 모두 저장되어 있어서 실제로는 저장된 블록을 호출 하여 사용 하는 갓이다.

파일의 입력을 이용하여 게임을 실행 시키기 위해, argc와 argv[] 이 두 argument값을 이용 하였다. 각각 조건을 만족하면 파일을 읽어 실행하는 모드 그리고 유저가 자유롭게 플레이 하는 모드를 지원하게 만들었다.

이 때 파일입력의 경우, 실시간으로 진행 과정을 볼 수 있게 하기 위해서 sleep이라는 함수를 이용하여, 천천히 게임을 진행하게 만들었고, 이로 인해 진행과정을 실시간으로 볼 수 있었다.

- 키보드 입력 시작시, 초기에 이름을 입력 받아, 이를 표시해주기 위해 전역변수에 동적할당을 해주어, 초기 윈도우 창에서 이름을 입력 받아 차후의 윈도우 창에 이를 표시 해주었고, 표시 후에 이를 delete 해주었다.

현실적 제한 요소 및 그 해결 방안

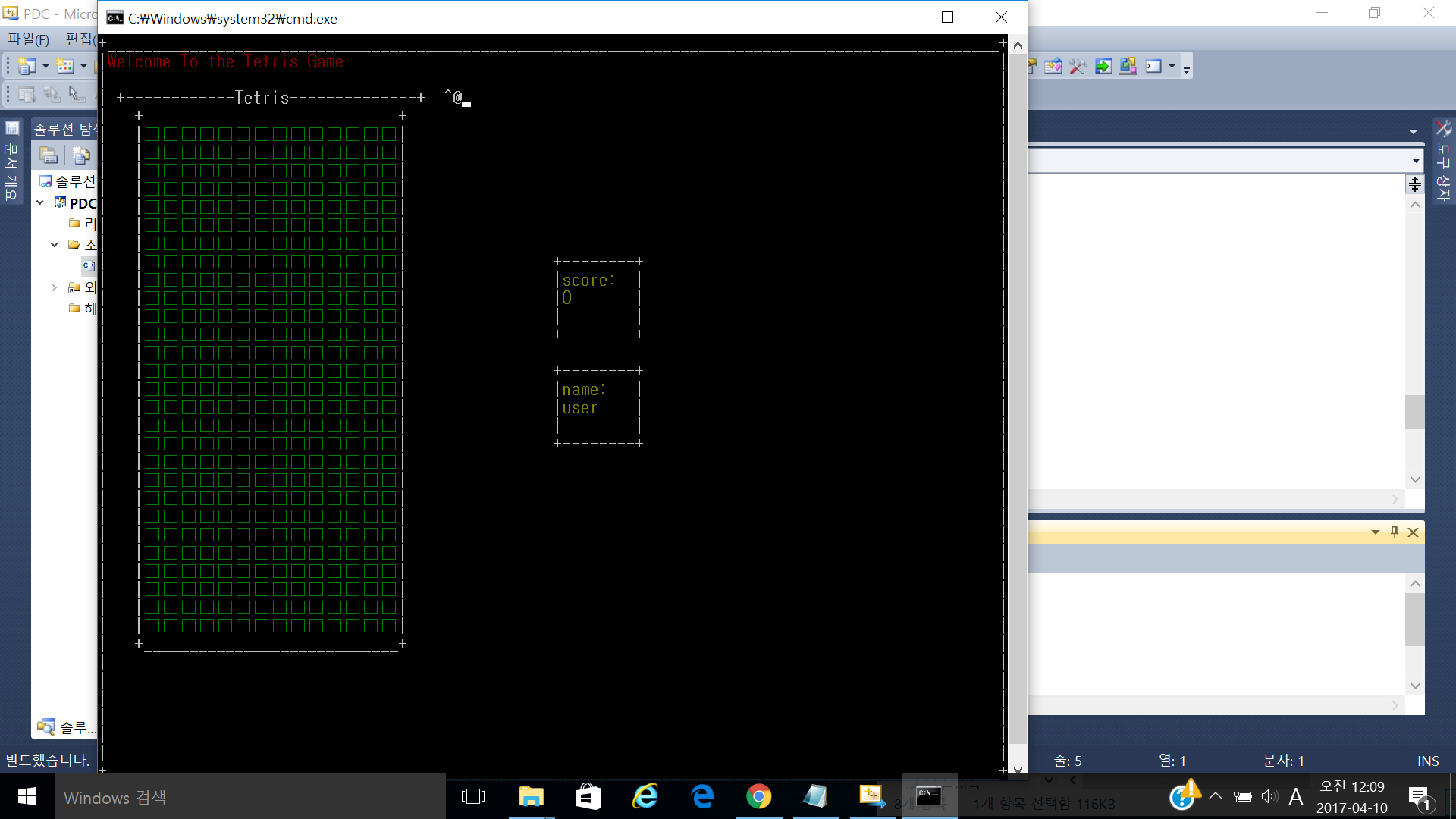
**- visual studio에 PDCurses를 실행하는 부분이 매우 어려웠다. 이부분은 첨부된 자료를 참고 하였지만, 해결이 안되어 결국 다른 웹에서 검색하여 해결 하였다.**

**- 윈도우 창에관한 개념이 부족하여서, 처음에는 윈도우 창속에 중복된 윈도우를 만들지 못하였다. 그러나 좌표에 관한 개념을 알게되니, 좌표를 설정하여 윈도우창이 만들어진다는 것을 알게 되어 이부분을 해결하였다.**

**- 겉으로 보기에는 매우 간단한 동작을 요구하는 것 같지만, 실상은 그렇지 않았다. 처음에 개념을 잡지 못했다. 어떻게 블록을 출력할 것인지, 블록의 경계는 어떻게 설정할 것인지에 대하여 해결하지 못했었다. 그래서 Helper를 찾아가 어떻게 구현해야 할지에 대한 전반적인 틀에 관한 설명을 들었지만, 여전히 해결되지 못한 부분은 많았다.**

**- 1차과제에서 앞으로 어떻게 구현해 나가야 할지에 대한, 대략적인 틀을 만들어 놓았지만, 이를 제대로 활용 할 수 없었다. 공부를 해보니 조사한 부분과 다른 점도 있었을 뿐더러, 아무것도 모른 채 시작을 해서 그런 지도 모르겠다.**

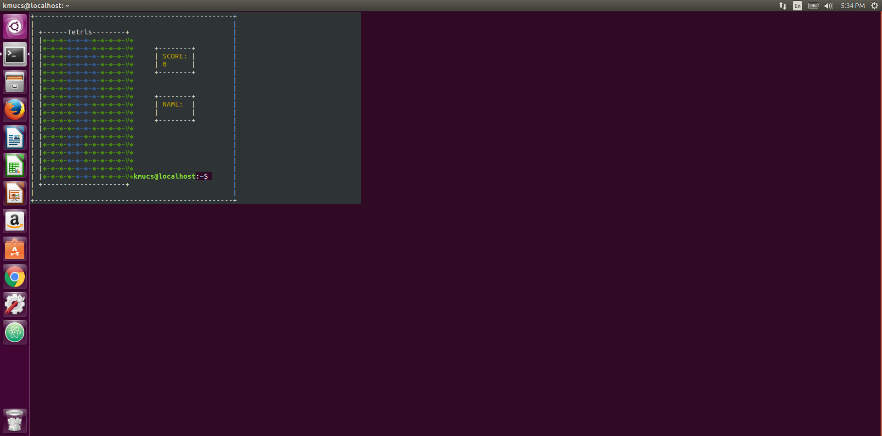
### 결과물 화면



기본적인 창의 사이즈를 조절하였고, 여기에 3개의 윈도우 창을 추가로 생성하여 각각 추가하였다. 첫번째 윈도우에는 배경을 설정하고, 여기에 ㅁ을 사용하여 칸을 구분 할 수 있게 하였다.

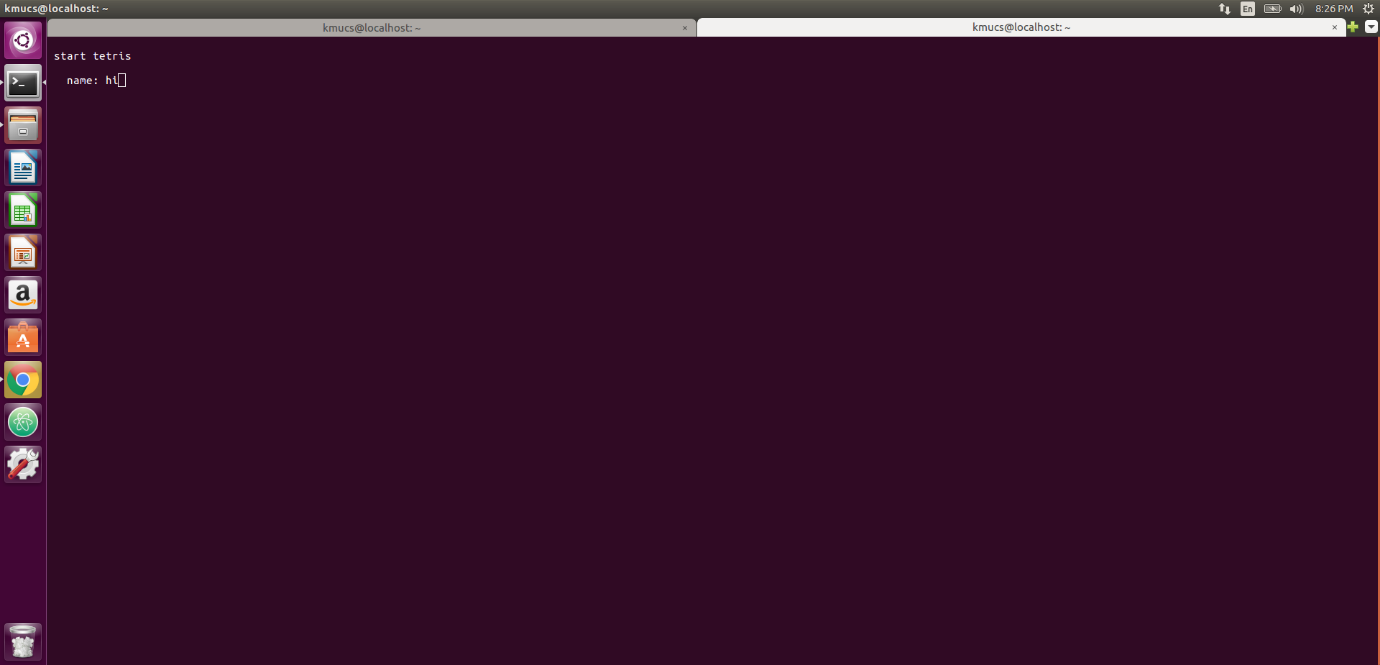
두번째 윈도우와 세번째 윈도우는 좌표 값과 출력되는 글자만 다를 뿐 그 외의 부분은 동일하다.

각각의 윈도우 그리고 배경에 다른 색을 지정하여 출력하였다.

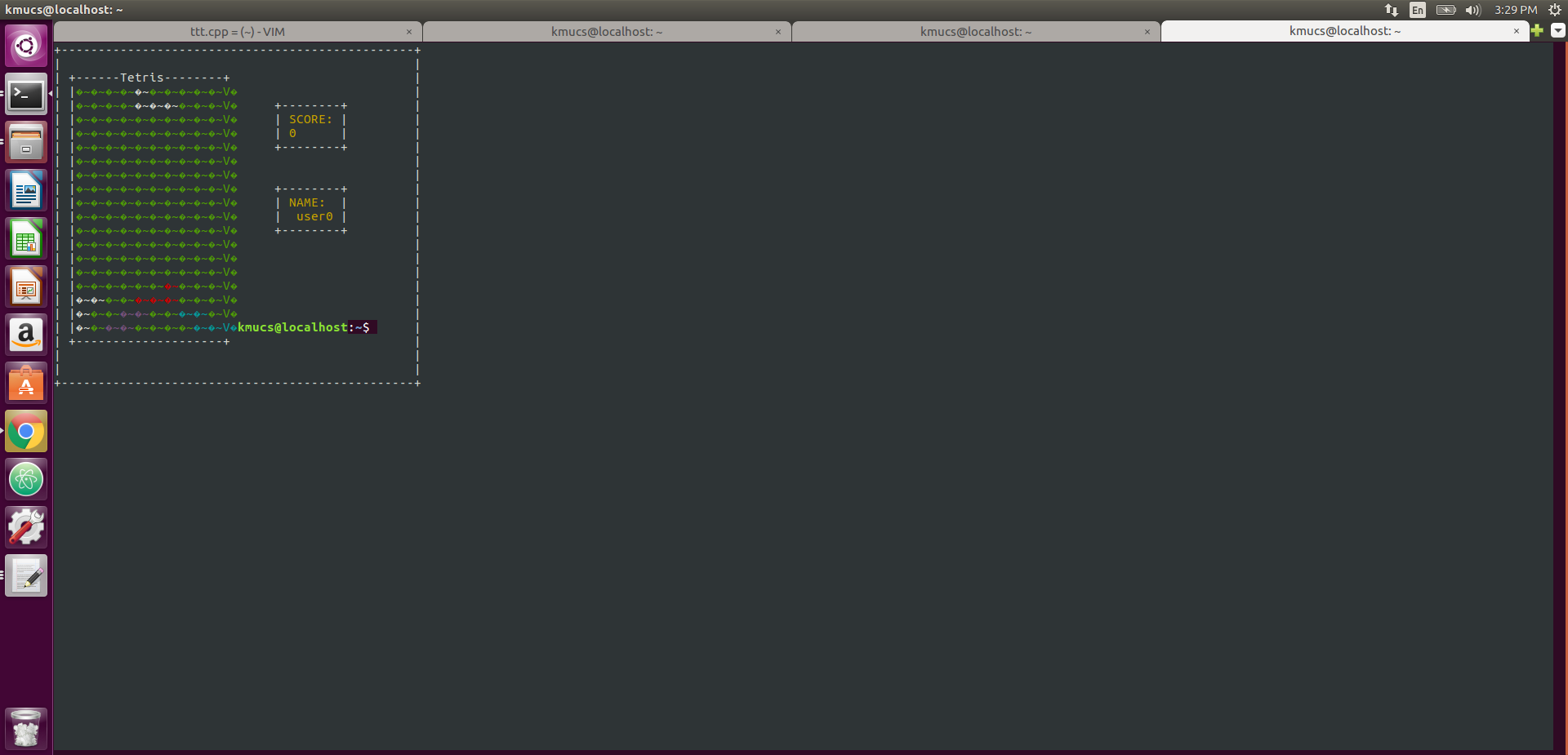


위의 알고리즘을 적용하면 다음과 같은 화면이 나오게 된다. 1차와 다른점이 있다면 리눅스 환경에서 코드를 작성하였고, 그로 인해 전반적인 변화가 많이 생기게 되었다.

3차 과제의 결과물은 다음과 같이 나타난다.



시작시 이름의 입력값을 받는 부분이 추가 되었다. 이때 이름을 입력하고, 엔터를 누르면 name을 담당하는 윈도우에 이름이 표시가 된다.



각 블록마다 색깔을 다르게 지정해 주었고, 블록이 바닥에 닿을 시 랜덤으로 새로운 블록이 생성된다. 2차 와 다른 점이 있다면 블록이 모두다 구현 되어 있다는 점이다.

## 기대효과 및 활용방안

- 지금까지 계획을 충실히 세웠고, 나아가 자료조사한 부분도 어느정도 쌓여 있기에, 이를 활용 한다면, 차후의 과제 수행에 있어서 수월함이 예상된다. 다만, 로직에 관한 설계는 아직 하지 않았기 때문에 이부분에 관해서는 조사가 필요하다. 그러나 기본적인 작동원리 그리고 블록에 대한 정의는 충실히 하였기에, 이 부분에 관해서는 안심이 된다.

- 조사를 한 결과와 이를 실제로 적용하는 괴리감이 상당하지만, 만약 조사자료가 없었더라면 과제를 수행할 수조차 없었을 것이다. 조사한 기본적인 알고리즘을 이용한다면, 많은 시행착오가 있긴 하겠지만, 이를 코드화 할 수 있다.

- 학교에서 과제를 해결할 수 있게 도와주는 헬퍼의 조언을 많이 얻었다. 확실히 도움을 받고 난 전후는 많은 차이가 났다. 우선 주어진 문제에 대한 개념을 잡을 수 있었을 뿐만 아니라, 구현의 측면에서도 도움을 많이 받았다.

- 기존의 클래스 설계도를 폐기하였다. 상속과 다형성을 이용하여 설계하긴 하였지만, 구체적으로 어떻게 구현방법이 너무나도 어렵기 때문이다. 그래서 최대한 간단한 개념 그리고 최대한 쉽게 다시 설계를 하였고 이를 이용하여 구현하였다.

# 자기평가

1차 과제를 수행하는 데에 있어 애로사항이 많았다. 정보가 부족했기, 책도 살펴보고 인터넷에서 검색을 많이 할 수 밖에 없었다. 다행히도 많은 시간을 공들여서 이번 1차과제는 간신히 완성 하였지만, 여전히 공부할 부분도, 차후 보완해야할 부분도 많이 남아있다. 아직 이 프로그램을 구동하기 위해 사용한 curses.h의 사용법 그리고 원리를 제대로 이해하지 못했을 뿐더러, 차후의 과제를 어떻게 실행해야 할지에 관하여 아직 아이디어가 없다. 다음 과제를 위해서 더 많은 시간을 투자하여, 이번 1차 과제를 보안 하는데 중점을 가질 뿐만 아니라, 남은 과제 또한 무사히 수행하는 것이 앞으로의 목표이다.

기획단계에서 착각으로 인하여, 설계를 다시 진행하였다. 차후에는 이번 일과 같은 상황이 발생하지 않도록, 조금 더 찾아보고 공부해야겠다.

2차 과제를 하는데 너무 어려워서 몇 번 이라도 포기하고 싶었다. 인터넷에 많은 코드들이 있지만 아무리 들여다 보더라도 그 코드를 이해할 수가 없었다. 그래서 결국 헬퍼를 찾아가서 전반적인 설명을 듣고 나서야, 작동원리를 알게 되었다. 하지만 원리를 알게 되었다고 해서, 구현은 또다른 문제였다. 무엇을 만들어야 할지는 알게 되었지만, 어떻게 할지에 관한 고민을 정말로 많이 했다. 이방법도 써보고 저 방법도 써보고 많이 시도를 해봤다. 결국 이런저런 사정으로 완벽하게 완성을 할 수는 없었다. 이부분은 정말로 아쉬웠지만, 3차과제에서는 2차과제를 더 개량하여 조건을 모두 충족하는 그런 테트리스를 만들고 싶다.

이번 3차과제에서는 지난번과 같은 실수를 하지 않았다. 시간을 넉넉하게 할당하여 일어날 수 있는 버그들을 잡으려고 노력을 했다. 그리고 최대한 노력하여 지난 과제에서는 구현하지 못했던 것들 또한 구현하기 위해 노력을 하였다. 그 결과 주어진 과제를 완벽히는 아니더라도 지난 2차과제에 비해서는 월등한 완성도를 보이는 결과물을 만들어 낼 수 있었다. 과제를 모두 구현하니 편안함을 느끼는 동시에 성취감을 느낄 수 있었다. 다만 아쉬운 점이 있다면, 스스로의 힘으로 100퍼센트 구현하지 못한점은 아쉽다. 그래도 해냈다는 부분에 만족한다. 차후에 이런 프로젝트에 맞닥뜨렸을 때, 포기하지 않고 시도하는 힘을 얻은 좋은 기회였다.

# 참고 문헌

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 번호 | 종류 | 제목 | 출처 | 발행년도 | 저자 | 기타 |
| 1 | 웹 | PDCurses | 위키백과 |  |  |  |
| 2 | 웹 | <http://kkikkodev.tistory.com/44> | 티스토리 |  |  |  |
| 3 | 웹 | https://ko.wikipedia.org/wiki/%ED%85%8C%ED%8A%B8%EB%A6%AC%EC%8A%A4 | 위키백과 |  |  |  |
| 4 | 웹 | http://blog.naver.com/dnpc7848/220367009691 | 블로그 |  |  |  |
|  |  |  |  |  |  |  |