**2021년 2학기**

**프로그래밍과 문제해결**

**Assignment #2**

**담당교수: 윤은영**

**학번: 20217016**

**학과: 무은재학부**

**이름: 최대현**

**POVIS ID: daehyeonchoi**

**명예서약(Honor Code)**

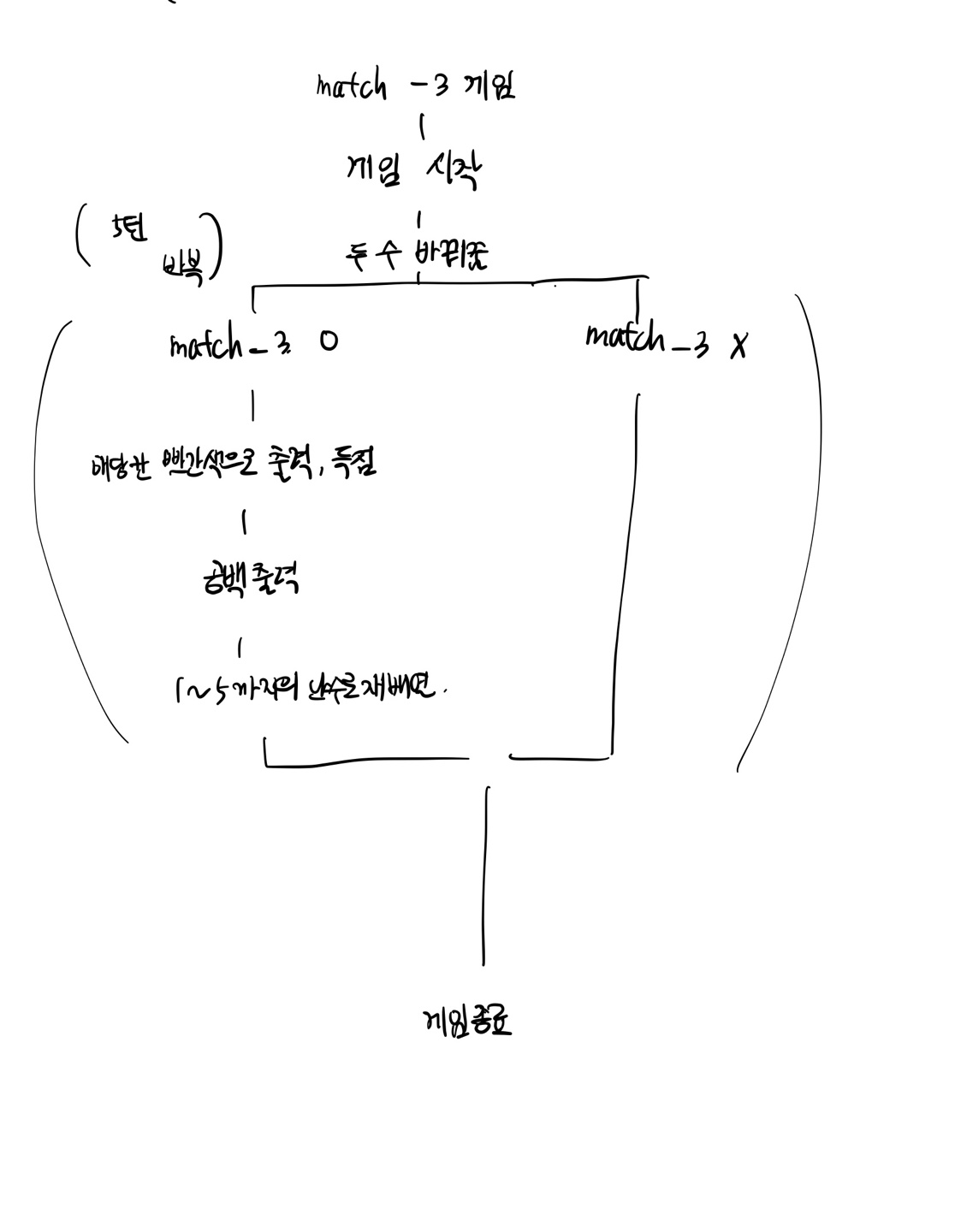
**“나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.”**

**프로그램 실행 방법**

(제작할 때 사용했던 Dev-C++ 환경 기준으로 설명) assn1.c 파일을 다운로드하고 실행한 뒤 F11키를 누르면 프로그램을 실행할 수 있다.

**Sturcture Chart**

이번 match3게임의 structure chart는 다음과 같다.



**알고리즘**

* **Psuedocode**

1. enter키를 누르면 게임이 시작된다.
2. 인접한 두 수의 인덱스를 입력받아 위치를 바꿔준다.

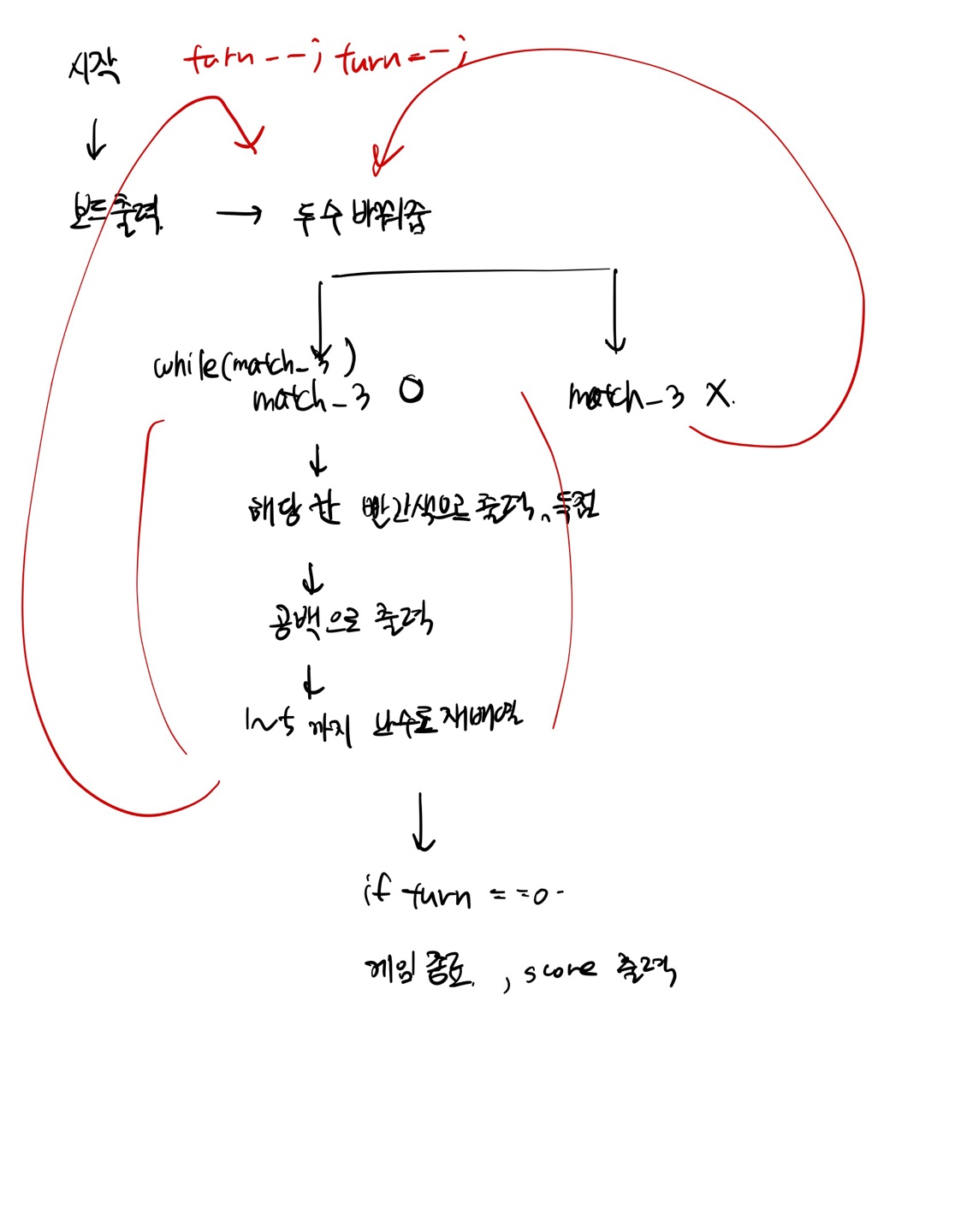
2-1. 입력받은 두 수의 인덱스가 범위를 벗어나면 다시 입력받는다.

2-2. 입력받은 두 수가 인접하지 않으면 다시 입력받는다.

1. Match\_3(같은 수가 3개 이상 연속 배열되어 있는 경우) 상황이 발생하면 그 만큼 득점하고, 그 부분을 다시 1~5 범위의 난수로 채워준다.
2. 5턴을 모두 진행했을 경우 게임이 종료되고 점수를 출력한다.

* **Flowchart**

이번 매치3 게임의 대략적인 Flowchart는 다음과 같다.



**함수 설명**

1. **void print\_board(int board[][BOARD\_SIZE], int match\_block[][BOARD\_SIZE], int score)**

* board를 출력해주는 함수이다. 빨간색으로 출력, 공백으로 출력하는 모든 기능을 조건문을 통해 한 번에 구현하였다.

1. **int check\_position(int board[][BOARD\_SIZE], int col1, int col2, int row1, int row2)**

* 사용자가 바꿔주고자 하는 두 값이 붙어있고, 배열 범위 내에 있는지 체크해주는 함수.이다. 2가지 값의 행과 열을 받아온다.

1. **void set\_color\_red()**

* 빨간색으로 출력할 수 있게 해주는 함수이다.

1. **void reset\_color()**

* 빨간색으로 바뀌었던 출력 색을 다시 되돌려주는 함수이다.

1. **void clean\_match\_block (int match\_block[][BOARD\_SIZE])**

* match\_block 배열 내의 모든 값을 0으로 초기화해주는 함수이다. 2중 for문으로 모든 인덱스에 들어있는 값을 0으로 초기화해준다. (다음 턴으로 넘어가기 위해)

1. **void swap(int\* a, int\* b)**

* 자주 사용하는, call-by-address 기법의 swap함수이다.

1. **void empty\_board(int board[][BOARD\_SIZE], int match\_block[][BOARD\_SIZE])**

* match\_3가 인식된 값들은 그 배열을 비워줘야 하므로, 그 과정을 구현하기 위해 이 함수를 호출할 경우 match\_block 내의 값이 0보다 큰 수이면 -1로 바꿔주었다. 그 뒤 print\_board에서 match\_block을 호출할 때 값이 -1이면 공백으로 출력하도록 처리하였다.

1. **int check\_match\_3(int board[][BOARD\_SIZE], int match\_block[][BOARD\_SIZE])**

* match\_3 상황을 인식하는 함수로, int형으로 선언하여 리턴값을 이번 라운드의 스코어 값으로 해 주었다. match\_3 상황이 발생하면 그에 해당하는 인덱스에 들어 있는 board 배열의 값을 match\_block 배열에 그대로 넣어주도록 설계했기 때문에 이 함수의 마지막 부분에 match\_block 함수값을 다 더해주면 그것이 곧 이 round에서 score 값을 의미한다. 이 함수의 리턴값이 0이라면 match\_3 상황이 없다는 것이므로 다음 턴으로 넘겨주도록 main함수를 구현했다.

1. **void rand\_board(int board[][BOARD\_SIZE], int match\_block[][BOARD\_SIZE])**

* 포인터를 사용하지 않고 구현하기 위해 void로 주어졌던 함수의 형을 int로 변환해서 사용하였다. 어느 칸에 있는지에 따라 다른 기능을 구현해야 하므로 user\_pos를 가져와서 switch-case문으로 user\_pos에 따라, 즉 현재 유저의 위치에 따라 다른 case들을 구현하였다. 각각의 케이스들에 return값을 넣어줌에 따라 game\_start 함수에서 play\_juru함수의 리턴값에 따라 각각 주량을 다르게 넣어주도록 설계하였다.

**프로그램 실행 방법 및 예제**

1. **초기 선택 메뉴**

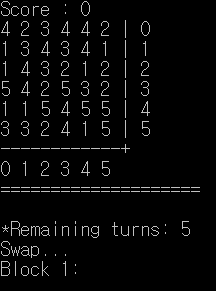
초기 선택 메뉴에서 enter키를 입력하면 넘어간다. enter키를 인식하려면 char형의 변수를 선언하여 입력받아야한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. **게임 시작**

enter키를 입력하면 게임이 시작되고, 현재 score(0으로 초기화)와 보드를 출력하고 남은 턴을 출력한다. 그 뒤 block 1의 행과 열 값을 입력받고, 정상적으로 입력되면 block 2의 행과 열 값을 입력받는다. 이 때 두 block은 가로 또는 세로로 붙어있어야 한다.



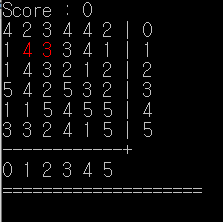
이 때 둘 중 하나의 값이라도 인덱스의 범위를 벗어나면 Out of range! 라는 문구를 출력하며 다시 입력받는다. 또한 두 값이 인접해있지 않는 경우에도 \*\*\*They are not adjacent! \*\*\*라는 문구를 출력하며 다시 입력받는다.

텍스트이(가) 표시된 사진

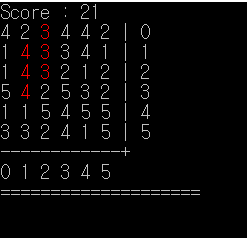
자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

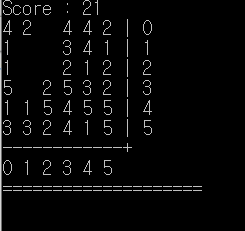
정상적으로 두 값이 입력됐다면, swap 함수를 통해 두 수의 위치를 서로 바꿔준다. 이 때 바꿔주었음을 알리기 위해 1초간 빨간색으로 글씨를 출력한다.



그 뒤 match\_3에 해당하는 값들이 있으면 1초간 빨간색으로 출력하는 동시에 득점하고, 점수판에 점수가 추가된다.



그 뒤, match\_3에 해당되는 위치를 공백 처리한 화면을 1초간 출력한다.



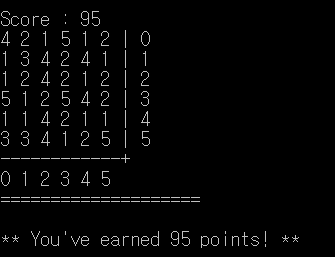
그 뒤, match\_3 상황이 없어질 때 까지 위 실행을 반복하고, match\_3 상황이 없어지면 턴이 하나 줄면서 다음 보드가 출력된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. **게임 종료**

게임이 5턴동안 진행되면 게임이 자동으로 종료되고, 그동안 획득한 점수를 출력하며 게임을 마치게 된다

.

**토론**

Match\_3를 인식하는 알고리즘이 담긴 함수인 check\_match\_3 함수를 구현할 때 알고리즘을 떠올리느라 시간을 많이 사용했는데, 생각보다 간단한 알고리즘으로 제작할 수 있었다. 또한 과제 설명 마지막 부분의 힌트대로 match\_block이라는 2차원 배열을 하나 더 생성했더니 과제를 수월하게 해결할 수 있었다.

**결론**

이번 assignment를 통하여 다양한 반환 형태를 가진 사용자 정의 함수를 제작해볼 수 있었고, 2차원 배열을 다루는 연습을 할 수 있었다.

**개선 방향**

사용자 정의 함수를 더 많이 사용해서 코드의 가독성을 높이고 코드를 더 간단하게 만들 수 있을 것 같아서 개선해보고 싶다.