과제 #6

HW1, HW2

과제 제출 마감: 11/19 24:00

조교 노인우, inwoo13@hanyang.ac.kr
조교 한중수, soohan@hanyang.ac.kr

Homework

- 과제 제출 마감: 11/19 24:00
- Gitlab repository에 “HW6” 폴더를 만든 후 진행
- 실습 서버 제출, 개인 PC 제출 중 편한 방법으로 제출
- 각 과제마다 HW6 폴더 내에 과제별 폴더를 만든 후 제출
- 채점 기준은 실습서버 환경에서 채점

Homework_01 – "오목 (Omok)"

- 19x19 바둑판에 흰/검은 알의 오목을 판단하는 프로그램 작성
- 작성 예시

```
#define BLACK -1
#define NOBODY 0
#define WHITE 1
#define GROUND_SIZE 19

class Omok {
    public:
        Omok() : width_(GROUND_SIZE), height_(GROUND_SIZE), turn_(NOBODY) {}

  // 번갈아 (x,y)에 돌을 놓음 (x,y = 0~18). 첫 수는 항상 흑돌로 가정.
  // 판 바깥쪽에 놓거나 이미 놓인 자리에 놓을 경우 NOBODY를 리턴.
  // 정상적인 경우 돌을 놓고 turn_을 턴에 맞게 세팅하고 리턴.
  // 다음 턴이 아닌, 현재 턴을 리턴 (처음 호출된 Put은 BLACK을 리턴하게 됨)
  int Put(int x, int y);

  // winner는 흑돌이 이긴 경우 BLACK으로, 백돌이 이긴 경우 WHITE로, 승부가 나지 않은 경우 NOBODY로 세팅
  void IsOmok(int* winner) const;
  int Turn() const { return turn_; }

  // 필요 시 함수를 추가 가능하나, width_, height_를 get하지 않고 class내에서 처리 후 출력까지 하는 설계를 권장
 private:
  int width_, height_;
  int turn_;                    // 마지막에 플레이 한 턴을 저장
  /* 멤버 변수를 추가. */
};

// 오목 판 출력
std::ostream& operator<<(std::ostream& os, const Omok& omok);
```

Homework_01 – "오목 (Omok)"

- 설계 시 고려사항
  - x, y좌표를 입력 받는 식으로 바둑돌을 놓음. (홀수턴은 검은돌, 짝수턴은 흰돌로 가정)
  - 입력은 순차적으로 승자가 나올 때까지 입력을 받고, 바둑판은 승자가 나왔을 때 프로그램 종료 직전 1회만 출력 (테스트 케이스에서 항상 승자 존재)
  - 검은돌, 흰돌은 영어 소문자인 o와 x로 표현
  - 바둑판은 '.'으로 표시하고 사이에 빈칸 한 칸씩 추가 (시각적 편의를 위함)
  - 입력을 무한히 받다가 승자가 나오면 Winner를 출력하고 프로그램 종료 (메시지는 입출력 양식 참고)
  - 같은 색의 바둑돌이 5개가 이어져 있는 경우만 승리로 예측. 6개는 인정 안됨
  - 이미 바둑돌이 놓여져 있는 위치나 바둑판 바깥쪽 위치가 입력되면 예외처리 (메시지는 입출력 양식 참고)
  - 바둑돌을 놓을 수 없다는 경고메시지만 출력해주고 턴은 그대로 유지

- 파일명: omok 폴더 내에
  (omok.h, omok.cc, omok_main.cc)
- 입력: 바둑돌을 놓을 위치인 x y 좌표
- 출력: 승부가 난 경우 19 x 19 바둑판을 표시하고 지금까지 놓인 바둑돌의 결과

Homework_01 – “오목 (Omok)”

- 입력 및 출력 예시

```
$ ./omok
O 9 9
X 9 8
O 10 9
X 8 9
O 8 8
  9 9
  Can not be placed there          // 19 19 밖(음수, 19 이상)에 놓아도 표시
X 7 7
O 10 10
X 10 7
O 11 11
X 7 10
O 12 12
0  ...................
1  ...................
2  ...................
3  ...................
4  ...................
5  ...................
6  ...................
7  .......x..x........
8  ........ox.........
9  .......xoo.........
10 .......x..o........
11 ..........o........
12 ...........o.......
13 ...................
14 ...................
15 ...................
16 ...................
17 ...................
18 ...................
Winner: Black player
$
```

10, 7

7, 7

...

12, 12

01234567890123456789

Homework_02 – "2차원 점 클래스 (2D point class)"

- 설계 시 고려사항
  - 2차원 벡터 연산을 구현한다
  - 다음의 명령을 처리한다
  - eval lhs operator rhs : Point class 에 대한 간단한 연산 결과를 출력
    - lhs나 rhs에 숫자가 올 수 있음 (Point(int c) constructor 사용)
    - 변수 이름 앞에 -가 붙을 수 있음 (x = (1,2), -x => (-1, -2))
  - set point x_val y_val : Point instance를 생성
  - quit : 프로그램 종료
- 파일명: point2d 폴더 내에
  (point2d.h, point2d.cc, point2d_main.cc)
- 입력 및 출력: 다음 슬라이드 참조

설계 예시 (참고용)
자유롭게 구현

```
struct Point {
  int x_, y_;                                   // 멤버 변수

  Point();
  Point(const Point& p);
  explicit Point(int c);
  Point(int x, int y);

  Point operator-();                            // 전위 - 연산자
};

Point operator+(const Point& lhs, const Point& rhs);
Point operator-(const Point& lhs, const Point& rhs);
Point operator*(const Point& lhs, const Point& rhs);
```

http://hashcode.co.kr/questions/325/c의-explicit-키워드는-뭘-하는-건가요
https://stackoverflow.com/questions/1711990/what-is-this-weird-colon-member-syntax-in-the-constructor

Homework_02 – "2차원 점 클래스 (2D point class)"

◆ 입력 및 출력 예시

```
$ ./point2d
set x 7 9          // Point x 를 (7, 9) 으로 초기화
set y 5 4
eval x + y         // x + y = (12, 13)
(12, 13)
eval x + 5         // x + (5, 5) = (12, 14)
(12, 14)
eval 3 - y         // (3, 3) - y = (-2, -1)
(-2, -1)
set z 1 2
eval x * z         // x * z = (7, 18)
(7, 18)
eval 1 * 2         // (1, 1) * (2, 2) = (2, 2)
(2, 2)
eval w * z         // w 가 정의되지 않았으므로 에러
input error
quit
```


Q&A