(선택) 텀 - 과제 001 - chess game - 체스 기물의 움직임

제출 마감일: 2023-04-07 23:59
업로드 가능한 파일 수: 4
제출 방식: 개인

주의

과제 032 은 keylog 분석 및 jplag 를 이용한 표절 검사가 상시 수행됩니다.

- 키로그 파일(p32.csv)에 소스 코드 이름이 Chess.cpp, main.cpp 등으로 제출 파일명과 일치하지 않으면 로그 점수가 계산되지 않습니다.
- 키로그 파일(p32.csv)에 키보드 입력 정보가 없는 경우 로그 점수가 계산되지 않습니다. (로그가 IdeState 나 Action 타입 밖에 없는 경우가 종종 있습니다.)
- 계속 잘 안될 때에는 초심으로 돌아가 activitytracker 를 clear 하고, 소스 파일명을 제출 파일명과 일치 시킨 후 직접 키보드로 코딩하는 것을 추천 드립니다.

목적

C++ 의 제어 구조문 (if, for, while 등)을 연습해 봅니다.

문제

체스 게임의 기물(piece) 를 적법한 위치로 이동하는 기능을 구현합니다.

비숍, 나이트, 폰, 룩의 이동에 대해서만 제한적으로 구현해 봅니다.

참고: https://www.chess.com/lessons/how-to-move-the-pieces

과제의 체스 게임 프로그램은 다음과 같은 순서로 동작합니다.

1. 체스판을 출력합니다. 체스판의 좌표는 x,y 로 표시하며 왼쪽 상단이 (0, 0) 입니다.
2. 체스판 위의 기물 중 하나를 x,y 로 선택합니다. **-1 -1 을 선택하면 프로그램을 종료합니다.**
3. 선택된 기물이 이동 가능한 좌표 (x, y) 를 모두 출력합니다. **x, y 오름차순으로 정렬하여 출력합니다.**
4. 이동 가능한 좌표 중 하나를 선택하여 기물을 이동합니다.
5. 1번부터 반복해서 수행합니다.

단계별 프로그램 실행 예제입니다.

```
  0  1  2  3  4  5  6  7
0 Rw Nw Bw Qw Kw Bw Nw Rw
1 Pw Pw Pw Pw Pw Pw Pw Pw
2
3
4
5
6 Pb Pb Pb Pb Pb Pb Pb Pb
7 Rb Nb Bb Qb Kb Bb Nb Rb
Select your piece (x y): 1 1
Legal moves: (2, 1) (3, 1)
Select a move (0-1): 1
  0  1  2  3  4  5  6  7
0 Rw Nw Bw Qw Kw Bw Nw Rw
1 Pw    Pw Pw Pw Pw Pw Pw
2
3    Pw
4
5
6 Pb Pb Pb Pb Pb Pb Pb Pb
7 Rb Nb Bb Qb Kb Bb Nb Rb
Select your piece (x y): 6 5
Legal moves: (4, 5) (5, 5)
Select a move (0-1): 0
  0  1  2  3  4  5  6  7
0 Rw Nw Bw Qw Kw Bw Nw Rw
1 Pw    Pw Pw Pw Pw Pw Pw
2
3    Pw
4                Pb
5
6 Pb Pb Pb Pb Pb    Pb Pb
7 Rb Nb Bb Qb Kb Bb Nb Rb
Select your piece (x y): -1 -1
```

입력

- 기물의 선택: x y 입력
- 이동 가능한 N개의 위치들 중 하나를 선택: 0 ~ N-1

출력

- 체스판의 현재 상태를 기물과 색상과 함께 출력함: Pb 는 폰(검은색) 을 의미함, Pw 는 폰(흰색) 을 의미함

<참고>

```
// chess.h
#include <vector>

const int BOARD_SIZE = 8;

// Piece types
enum Piece {
  EMPTY,
  PAWN,
  ROOK,
  KNIGHT,
  BISHOP,
  QUEEN,
  KING
};

// Colors
enum Color {
  BLANK,
  WHITE,
  BLACK
};

// Struct for a chess piece
struct ChessPiece {
  Piece piece;
  Color color;
};

typedef ChessPiece ChessBoard[BOARD_SIZE][BOARD_SIZE];

/**
 * @brief Prints the chess board to the console.
 *
 * @param board The board to print.
 */
void print_board(ChessBoard const board);

/**
 * @brief Finds all the legal moves for a bishop on the chess board.
 *
 * @param board The board to search for legal moves.
 * @param x The x-coordinate of the bishop.
 * @param y The y-coordinate of the bishop.
 * @param color The color of the bishop.
 * @return A vector containing all the legal moves for the bishop.
 * @ref https://www.chess.com/lessons/how-to-move-the-pieces
 */
std::vector<std::pair<int, int>> find_bishop_moves(ChessBoard board, int x, int y, Color color) ;

/**
 * @brief Finds all the legal moves for a knight on the chess board.
 *
 * @param board The board to search for legal moves.
```

```
 * @param board The board to search for legal moves.
 * @param x The x-coordinate of the knight.
 * @param y The y-coordinate of the knight.
 * @param color The color of the knight.
 * @return A vector containing all the legal moves for the knight.
 * @ref https://www.chess.com/lessons/how-to-move-the-pieces
 */
std::vector<std::pair<int, int>> find_knight_moves(ChessBoard board, int x, int y, Color color);

/**
 * @brief Finds all the legal moves for a pawn on the chess board.
 *
 * @param board The board to search for legal moves.
 * @param x The x-coordinate of the pawn.
 * @param y The y-coordinate of the pawn.
 * @param color The color of the pawn.
 * @return A vector containing all the legal moves for the pawn.
 * @ref https://www.chess.com/lessons/how-to-move-the-pieces
 */
std::vector<std::pair<int, int>> find_pawn_moves(ChessBoard board,int x, int y, Color color);

/**
 * @brief Finds all the legal moves for a rook on the chess board.
 *
 * @param board The board to search for legal moves.
 * @param y The y-coordinate of the rook.
 * @param x The x-coordinate of the rook.
 * @param color The color of the rook.
 * @return A vector containing all the legal moves for the rook.
 * @ref https://www.chess.com/lessons/how-to-move-the-pieces
 */
std::vector<std::pair<int, int>> find_rook_moves(ChessBoard board,int x, int y, Color color);

#endif //INC_2022_WINTER_QUIZ_CHESS_H_



//chess.cpp
// implement the functions in chess.h



// chess_main.cpp

bool make_move(ChessBoard board, int from_x, int from_y) {
  ChessPiece piece = board[from_x][from_y];

  std::vector<std::pair<int, int>> legal_moves;
  switch (piece.piece) {
     case PAWN: {
     // implement your code
     ....
}

int main() {
  ChessBoard board = {
      {{ROOK,   WHITE}, {KNIGHT, WHITE}, {BISHOP, WHITE}, {QUEEN, WHITE},{KING,  WHITE}, {BISHOP, WHITE}, {KNIGHT, WHITE}, {ROOK,   WHITE}},
      {{PAWN,   WHITE}, {PAWN,   WHITE}, {PAWN,   WHITE}, {PAWN,  WHITE}, {PAWN,  WHITE}, {PAWN,   WHITE}, {PAWN,   WHITE}, {PAWN,   WHITE}},
      {{EMPTY,  BLANK}, {EMPTY,  BLANK}, {EMPTY,  BLANK}, {EMPTY, BLANK}, {EMPTY, BLANK}, {EMPTY,  BLANK}, {EMPTY,  BLANK}, {EMPTY,  BLANK}},
      {{EMPTY,  BLANK}, {EMPTY,  BLANK}, {EMPTY,  BLANK}, {EMPTY, BLANK}, {EMPTY, BLANK}, {EMPTY,  BLANK}, {EMPTY,  BLANK}, {EMPTY,  BLANK}},
      {{EMPTY,  BLANK}, {EMPTY,  BLANK}, {EMPTY,  BLANK}, {EMPTY, BLANK}, {EMPTY, BLANK}, {EMPTY,  BLANK}, {EMPTY,  BLANK}, {EMPTY,  BLANK}},
      {{EMPTY,  BLANK}, {EMPTY,  BLANK}, {EMPTY,  BLANK}, {EMPTY, BLANK}, {EMPTY, BLANK}, {EMPTY,  BLANK}, {EMPTY,  BLANK}, {EMPTY,  BLANK}},
      {{PAWN,   BLACK}, {PAWN,   BLACK}, {PAWN,   BLACK}, {PAWN,  BLACK}, {PAWN,  BLACK}, {PAWN,   BLACK}, {PAWN,   BLACK}, {PAWN,   BLACK}},
      {{ROOK,   BLACK}, {KNIGHT, BLACK}, {BISHOP, BLACK}, {QUEEN, BLACK}, {KING,  BLACK}, {BISHOP, BLACK}, {KNIGHT, BLACK}, {ROOK,   BLACK}}
  };

  print_board(board);

  while (true) {
    int from_x, from_y;
```

```
    std::cout << "Select your piece (x y): ";
    std::cin >> from_x >> from_y;
    std::cin.ignore();
    if (from_x == -1 && from_y == -1) break;

    if (!make_move(board, from_x, from_y)) {
      std::cout << "Illegal move!\n";
      continue;
    }

    print_board(board);
  }

  return 0;
}
```

제출파일

chess.h

chess.cpp

chess_main.cpp

p32.csv