Stratego

Start :

First of all, make sure that you're running it in the terminal. Then run the subdir called tui. Same goes for the test, run test subdir instead of tui.

Gameplay TUI:

Before the game starts it will asks you if all the cases should be visible [y/n] answer is needed.

```
Do you want to use the cheat mode? [y/n]
```

Then the program goes through swapping mode just after the initialization of random pieces into the board and the file.

```
########PLAYER 1########


     A B C D E F G H I J
  ----------------------
0 |  ? ? ? ? ? ? ? ? ? ?
1 |  ? ? ? ? ? ? ? ? ? ?
2 |  ? ? ? ? ? ? ? ? ? ?
3 |  ? ? ? ? ? ? ? ? ? ?
4 |      W W     W W
5 |      W W     W W
6 |  1 2 7 4 0 6 2 7 B F
7 |  5 5 B 2 4 B 6 6 9 B
8 |  8 4 5 5 4 B 1 B 1 3
9 |  2 1 1 2 1 3 1 1 3 3

To swap pieces, type the two piece positions like this : A7 C7
Type END to start the game
```

After the swapping mode has ended, the game starts. Player one is choosed by default and have to choose it's position to move from, direction of the move and the distance (only for scout). Ex.: J6 TOP or if there's a scout: J6 TOP 2

```
########PLAYER 1########


     A B C D E F G H I J
  ----------------------
0 |  ? ? ? ? ? ? ? ? ? ?
1 |  ? ? ? ? ? ? ? ? ? ?
2 |  ? ? ? ? ? ? ? ? ? ?
3 |  ? ? ? ? ? ? ? ? ? ?
```

```
4 |       W W     W W
5 |       W W     W W
6 |   1 2 7 4 0 6 2 7 B F
7 |   5 5 B 2 4 B 6 6 9 B
8 |   8 4 5 5 4 B 1 B 1 3
9 |   2 1 1 2 1 3 1 1 3 3

Insert position from where you'd like to move, then direction, then the distance(optional)
Example: A7 TOP
```

So the game continues on ;)

Technical informations :

The project respects the MVP architecture structure.

Warnings

There's only 1 warning with parameter in no use, in the class Board. This is made on purpose, because we needed a second constructor that does not initialize the board for tests only.

View part :

The project have got two views and controlers : one for graphical interface and an other one for console interface.

TUI part:

Facade part :

The Facade manage the players turns, the states and interactions between GUI/CLI Views and Board model.

Model part :

Board :

Board structure : The board manage a 2D std::array of std::optional Pieces (`std::array<std::array<std::optional<Piece>>>`). With this structure, we can easy-way detect if a board case is empty or not and access to a case without loop.

Forbidden locations : To manage forbidden cases (e.g. : water cases), we will just put a Piece with 'W' symbol.

Movables/unmovables pieces : To know if a piece is movable, we just have to check if piece symbol is an alphabetic or numeric char(alphabetics are unmovables and numerics movables).

Moves : Each turn, a player chooses a piece and a direction. If the piece can move more than one case (scout), the player add a move distance.

Enemy detection : After player chooses a move, the program will check (case by case if the move distance is bigger than 1) if a ennemy is on a case (by calling x and y element in board array), no more move is allowed for this turn and the program check wich piece(s) have to leave the board.

Game over : Game is over when the flag piece is catch or there's no pieces to move.

Project structure:

```
   .gitignore
   README.md
   UML - stratego.mdj

  Stratego
       config.pri
       main.cpp
       Position.cpp
       Position.hpp
       Stratego.pro

     Model
          Board.cpp
          Board.hpp
          Direction.hpp
          Facade.cpp
          Facade.hpp
          Model.pro
          Piece.cpp
          Piece.hpp
          Position.cpp
          Position.hpp
          State.hpp

     tests
          catch.hpp
          main.cpp
          tests.pro
     test_models.cpp

tui
     Controller.cpp
     Controller.h
     main.cpp
     tui.pro
     view.cpp
     view.h
```