TPP2020-HW1-40747024S 于子緯

main.cpp

```
#include "robot.h"
#include "maze.h"

#include <iostream>
#include <vector>
using namespace std;

int main() {

    // read input
    int w, h;
    long long n;
    cin >> w >> h >> n;
    vector<string> temp_maze(h);
    pair<int,int> st;
    for ( int i=0; i<h; ++i ) {
        cin >> temp_maze[i];
        for ( int j=0; j<w; ++j )
            if ( temp_maze[i][j] == 'O' )
                st = make_pair(i, j);
    }

    // initialize object with constructor
    Maze maze = Maze(w, h, temp_maze);
    Robot robot = Robot(n, st);

    // start moving & get terminal location
    robot.printLocation(robot.go(maze));

    return 0;
}
```

maze.h

```
#ifndef MAZE__
#define MAZE__

#include <vector>
#include <string>
using namespace std;

class Maze {
public:
    Maze(int w, int h, vector<string> mz):width(w), height(h), maze(mz) {}
    bool isObstacle(pair<int,int> loc) const;
private:
    const int width, height;
    const vector<string> maze;
};

#endif
```

maze.cpp

```
#include "maze.h"

// check whether this location is walkable or not
bool Maze::isObstacle(pair<int,int> loc) const {
    if ( loc.first < 0 || height <= loc.first )
        return false;
    if ( loc.second < 0 || width <= loc.second )
        return false;
    if ( maze[loc.first][loc.second] == '#' )
        return false;
    return true;
}
```

robot.h

```
#ifndef ROBOT__
#define ROBOT__

#include <iostream>
#include <vector>
#include <map>
using namespace std;

class Maze;

class Robot {
public:
    Robot(long long _step, pair<int,int> _start) {
        step = _step;
        location = _start;
        direction = 0;
    }
    pair<int,int> go(const Maze maze);
    void printLocation(pair<int,int> loc) {
        cout<<loc.second<<' '<<loc.first<<'\n'; }
private:
    // 0: up, 1: right, 2: down, 3: left
    int direction;
    const int dv[4] = {-1, 0, 1, 0}; // vertical delta
    const int dh[4] = {0, 1, 0, -1}; // horizontal delta
    long long step;
    pair<int,int> location;
    vector<pair<int,int>> history;
    map<pair<int,int>,int> stamp[4];
};

#endif
```

robot.cpp

```
#include "robot.h"
#include "maze.h"

// start moving until using up all the steps or in a loop.
pair<int,int> Robot::go(const Maze maze) {
    while ( step > 0 ) {
        /*
            if I've visit this location & the direction I face are both same
            , indicating I'm in a loop.
        */
        if ( stamp[direction][location] > 0 ) {
            // find the terminal location with mod.
            int cycle_length = history.size() - stamp[direction][location] + 1;
            return history[stamp[direction][location]-1 + step%cycle_length];
        }
        // push location into my history.
        history.emplace_back(location);
        /*
            stamp:
            stores index(+1) which the location & the direction at in history.
        */
        stamp[direction][location] = history.size();
        // trying which direction can I move.
        for ( int i=0; i<4; ++i ) {
            int new_direction = (direction + i) % 4;
            if ( maze.isObstacle(make_pair(location.first+dv[new_direction],
location.second+dh[new_direction])) ) {
                direction = new_direction;
                location.first += dv[direction];
                location.second += dh[direction];
                break;
            }
        }
        step--;
    }
    return location;
}
```