OOP2024 Refactor

B12902046 廖昀陽

March 31, 2024

Information

Problem

50260. L-Shaped Bricks in a Square Grid

Initial Submission

Ugly Code

Git Repository For This Homework

pp1457/Refactor

1 Refactor With Function

First, I realize there's no need to sort the corner point, I can store the entire grid marking which point is a corner point, and visit them in order. Then I divide the job into several functions, the "input" part is responsible for handling input, and the "identify" part will check the type and size of that L-shaped Brick. There' s another function called "available" for the "identify" part to check whether a square is part of a brick.

main

```
int main() {
    int n, m;
    int grid[SIZE][SIZE];
    bool isCorner[SIZE][SIZE] {{0}};

    Input(n, m, grid, isCorner);

    for (int i = 0; i < n; i++) for (int j = 0; j < n; j++)
        if (isCorner[i][j])
            identify(i, j, grid);
}
```

2 Refactor With Structure

In this version, I wrap the information about the grid into a structure, including the size, the number of corner points, and the position of each brick and corner point.

grid.h

```
#define SIZE 105
struct Grid {
    int n, m;
    bool notEmpty[SIZE][SIZE] {{0}};
    bool isCorner[SIZE][SIZE] {{0}};
};
```

main

```
int main() {
    Grid grid;
    Input(grid);

    for (int i = 0; i < grid.n; i++) for (int j = 0; j < grid.n; j++)
        if (grid.isCorner[i][j])
            identify(i, j, grid);
}
```

3 Refactor With All I Know

In this final version, I used vector instead of array for flexibility, and created two classes called "Grid" and "Coordinate".
"Grid" will handle the input, store the state of the grid, and calculate the answer, and "Coordinate" is responsible for the movement of points.

coordinate.h

```
const int dx[4] = {-1, 0, 1, 0}, dy[4] = {0, -1, 0, 1};

class Coordinate {
  private:
    int x, y;
  public:
    Coordinate(int x, int y): x {x}, y {y} {}
    Coordinate& move(int direction);
    int getX();
    int getY();
};
```

grid.h

```
#include "coordinate.h"
#include <vector>

using namespace std;

class Grid {
  private:
    int n, m;
    vector<vector<bool>> notEmpty, isCorner;
    void identify(Coordinate c);
    bool valid(Coordinate c);
  public:
    Grid();
    void input();
    void work();
};
```

main.cpp

```
#include "grid.h"
using namespace std;

int main() {
    Grid grid;
    grid.input();
    grid.work();
}
```