Team Notebook

August 5, 2023

Contents

1 main

```
#include <iostream>
#include <cmath>
#include <queue>
#include <stack>
#include <set>
#include <map>
#include <unordered_set>
#include <unordered_map>
#include <vector>
#include <iterator>
#include <algorithm>
#include <string>
#include <sstream>
typedef long l;
typedef long long ll;
typedef unsigned long ul;
typedef unsigned long long ull;
typedef long double ld;
#define pi M_PI
#define pb push_back
#define loop(i,n) for(long long i=0; i<n;i++)
#define in insert
using namespace std;

vector<vector<char>> box;
vector<char> v;
struct coord {
    int x = 0;
    int y = 0;
};

void OutputBox() {
    for(vector<char> vec: box) {
        for (char c: vec) {
            cout << c;
        }
        cout << endl;
    }
}

// Shifts all values in a vector to the right (it loops)
template<typename T>
void ShiftRight(vector<T>& vec) {
    T last = vec.back();
    for (int i=vec.size()-1; i > 0; i--) {
        vec[i] = vec[i-1];
    }
    vec[0] = last;
}

// Shifts all values in a vector to the left (it loops)
template<typename T>
void ShiftLeft(vector<T>& vec) {
    T first = vec.at(0);
    for (int i=0; i < vec.size()-1; i++) {
        vec[i] = vec[i+1];
    }
    vec[vec.size()-1] = first;
}

coord Move(coord co, string dir) {
    coord ne = co;
    if (dir == "down")
        ne.y++;
    else if (dir == "up")
        ne.y--;
    else if (dir == "left")
        ne.x--;
    else if (dir == "right")
        ne.x++;

    // Shift Right
    if (ne.x < 0) {
        for (int i=0; i < box.size(); i++) {
            ShiftRight(box.at(i));
        }
        ne.x++;
    }
    // Shift Left
    else if (ne.x > box.at(0).size() - 1) {
        for (int i=0; i < box.size(); i++) {
            ShiftLeft(box.at(i));
        }
        ne.x--;
    }
    // Shift Down
    else if (ne.y < 0) {
        ShiftRight(box);
        ne.y++;
    }
    // Shift Up
    else if (ne.y > box.size() - 1) {
        ShiftLeft(box);
        ne.y--;
    }

    return ne;
}

void BindBox() {
    for (int i=0; i < box.size(); i++) {
        box.at(i).insert(box.at(i).begin(), '#');
        box.at(i).pb('#');
    }
    vector<char> vec;
    vec.resize(box.at(0).size(), '#');
    box.insert(box.begin(), vec);
    box.pb(vec);
}

int main() {
    string input;
    vector<string> directions;
    int top_box=0, bottom_box=0, left_box=0, right_box=0;
    int x=0, y=0;

    // Input
    while (cin >> input) {
        directions.pb(input);

        if (input == "down")
            y--;
        else if (input == "up")
            y++;
        else if (input == "left")
            x--;
        else if (input == "right")
            x++;

        // Keeps track of rectange dimensions
        top_box = max(top_box, y);
        bottom_box = min(bottom_box, y);
        left_box = min(left_box, x);
        right_box = max(right_box, x);
    }

    // Box Dimensions (Debug Stuff)
    //  cout << "x: " << x << " y: " << y << endl;
    //  cout << "top: " << top_box << " bottom: " <<
         bottom_box << endl;
    //  cout << "left: " << left_box << " right: " <<
         right_box << endl;

    // Create Initial Box Vector
    v.resize( (right_box - left_box + 1), ' ');
    box.resize((top_box - bottom_box + 1) , v);

    // Trace Path
```

```
coord co;
box[co.x][co.y] = 'S';
for (int i=0; i < directions.size(); i++) {
    string dir = directions.at(i);
    co = Move(co, dir);
    if (box[co.y][co.x] != 'S')
        box[co.y][co.x] = '*';
}
box[co.y][co.x] = 'E';
BindBox();
    OutputBox();

    return 0;
}
```