UVA118

題目

Robotics, robot motion planning, and machine learning are areas that cross the boundaries of many of the subdisciplines that comprise Computer Science: artificial intelligence, algorithms and complexity, electrical and mechanical engineering to name a few. In addition, robots as “turtles” (inspired by work by Papert, Abelson, and diSessa) and as “beeper-pickers” (inspired by work by Pattis) have been studied and used by students as an introduction to programming for many years.

機器人技術、機器人運動規劃和機器學習是跨越許多學科界限的領域計算機科學的子學科：人工智能、算法和復雜性，電氣和機械工程僅舉幾例。 此外，機器人就像“海龜”（靈感來自Papert、Abelson 和 diSessa 的工作）和“蜂鳴器拾取器”（受 Pattis 的工作啟發）多年來，學生們一直在研究和使用它作為編程入門。

題目

This problem involves determining the position of a robot exploring a pre-Columbian flat world.

Given the dimensions of a rectangular grid and a sequence of robot positions and instructions, you are to write a program that determines for each sequence of robot positions and instructions the final position of the robot.

該問題涉及確定探索前哥倫布時期平坦世界的機器人的位置。

給你一塊矩形土地的長寬，再依序給定每個機器人的初始位置狀況及一連串的指令集，你必須用你的程式求出每個機器人最後的位置狀況。

題目

A robot position consists of a grid coordinate (a pair of integers: x-coordinate followed by ycoordinate) and an orientation (N,S,E,W for north, south, east, and west). A robot instruction is a string of the letters ‘L’, ‘R’, and ‘F’ which represent, respectively, the instructions:

- Left: the robot turns left 90 degrees and remains on the current grid point.
- Right: the robot turns right 90 degrees and remains on the current grid point.
- Forward: the robot moves forward one grid point in the direction of the current orientation and mantains the same orientation.

一個機器人的位置狀況包括了其坐標（ x 坐標， y 坐標），和它面向的方向（用 N , S , E , W 來分別代表北、南、東、西）。至於一個機器人所收到的指令集，是一個由字母 ' L ' ， ' R ' ，和 ' F ' 所構成的字串，其分別代表：

- 左轉（ Left ）：機器人在原地往左轉 90 度。
- 右轉（ Right) : 機器人在原地往右轉 90 度。
- 前進（ Forward) : 機器人往其面向的方向向前走一格，且不改變其面向之方向

題目

The direction North corresponds to the direction from grid point (x, y) to grid point (x, y + 1). Since the grid is rectangular and bounded, a robot that moves “off” an edge of the grid is lost forever. However, lost robots leave a robot “scent” that prohibits future robots from dropping off the world at the same grid point. The scent is left at the last grid position the robot occupied before disappearing over the edge. An instruction to move “off” the world from a grid point from which a robot has been previously lost is simply ignored by the current robot.

從坐標 (x,y) 走至 (x,y+1) 的這個方向我們定義為北方。因為此矩形土地是有邊界的，所以一旦一個機器人走出邊界掉落下去，就相當於永遠消失了。不過這個掉下去的機器人會留下「標記 (scent) 」，提醒以後的機器人，避免他們從同一個地方掉下去。掉下去的機器人會把標記，留在他掉落之前所在的最後一個坐標點。所以對於以後的機器人，當他正位在有標記的地方時，這個機器人就會忽略會讓他掉下去的指令。

輸入

The first line of input is the upper-right coordinates of the rectangular world, the lower-left coordinates are assumed to be 0,0. The remaining input consists of a sequence of robot positions and instructions (two lines per robot). A position consists of two integers specifying the initial coordinates of the robot and an orientation (N,S,E,W), all separated by white space on one line. A robot instruction is a string of the letters 'L', 'R', and 'F' on one line. Each robot is processed sequentially, i.e., finishes executing the robot instructions before the next robot begins execution. Input is terminated by end-of-file. You may assume that all initial robot positions are within the bounds of the specified grid. The maximum value for any coordinate is 50. All instruction strings will be less than 100 characters in length.

輸入裡的第一列有2個正整數，代表這個矩形世界右上角頂點的坐標，其中假設這個世界的左下角頂點坐標為 (0 , 0)。接下來是若干組有關機器人的初始位置狀況和指令集，每個機器人2列。第一列為位置狀況，包括了兩個整數和一個字元（ N , S , E 或 W ），代表機器人初始的位置坐標以及機器人最初所面對的方向。第二列則是指令集，是一個由 'L' ， 'R' 和 'F' 所組成的字串。輸入以 end-of-file 作為結束。各機器人是依序動作的，也就是說，直到一個機器人作完他全部的動作，下一個機器人才會開始動作。所有機器人的初始位置皆會在矩形土地上，不會落在外面。任何坐標的最大值皆不會超過 50 。每個指令集的長度皆不會超過 100 個字元長。

輸出

For each robot position/instruction in the input, the output should indicate the final grid position and orientation of the robot. If a robot falls off the edge of the grid the word 'LOST' should be printed after the position and orientation.

對於每一個機器人，你都必須輸出其最後所在的坐標和面對的方向。如果一個機器人會掉落出此世界外，你必須先輸出他在掉落前，最後的所在位置和面對的方向，再多加一個字：LOST 。

範例測資

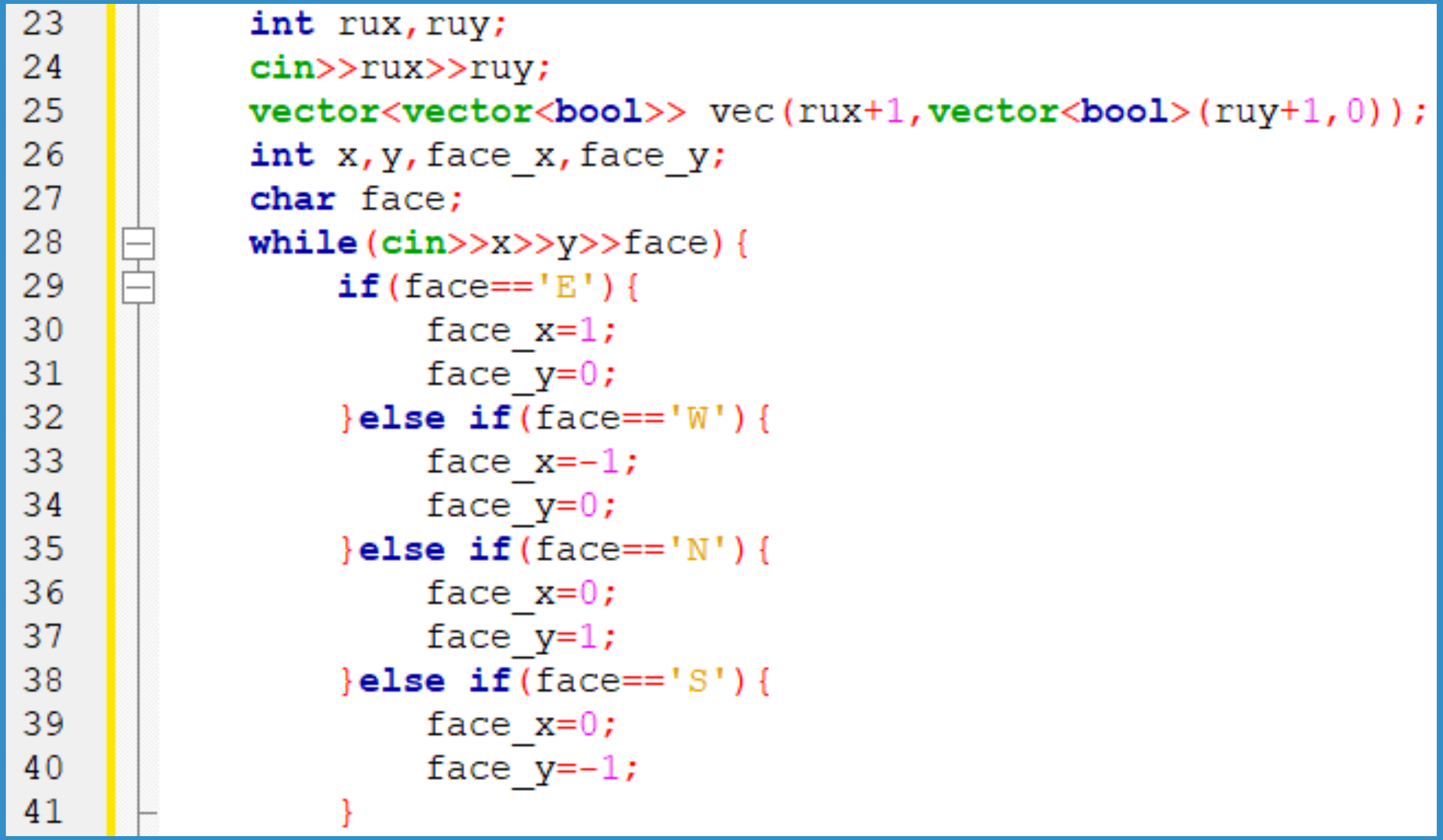
Input：

```
5 3
1 1 E
RFRFRFRF
3 2 N
FRRFLLFFRRFLL
0 3 W
LLFFFLFLFL
```

Output：

```
1 1 E
3 3 N LOST
2 3 S
```

程式碼說明

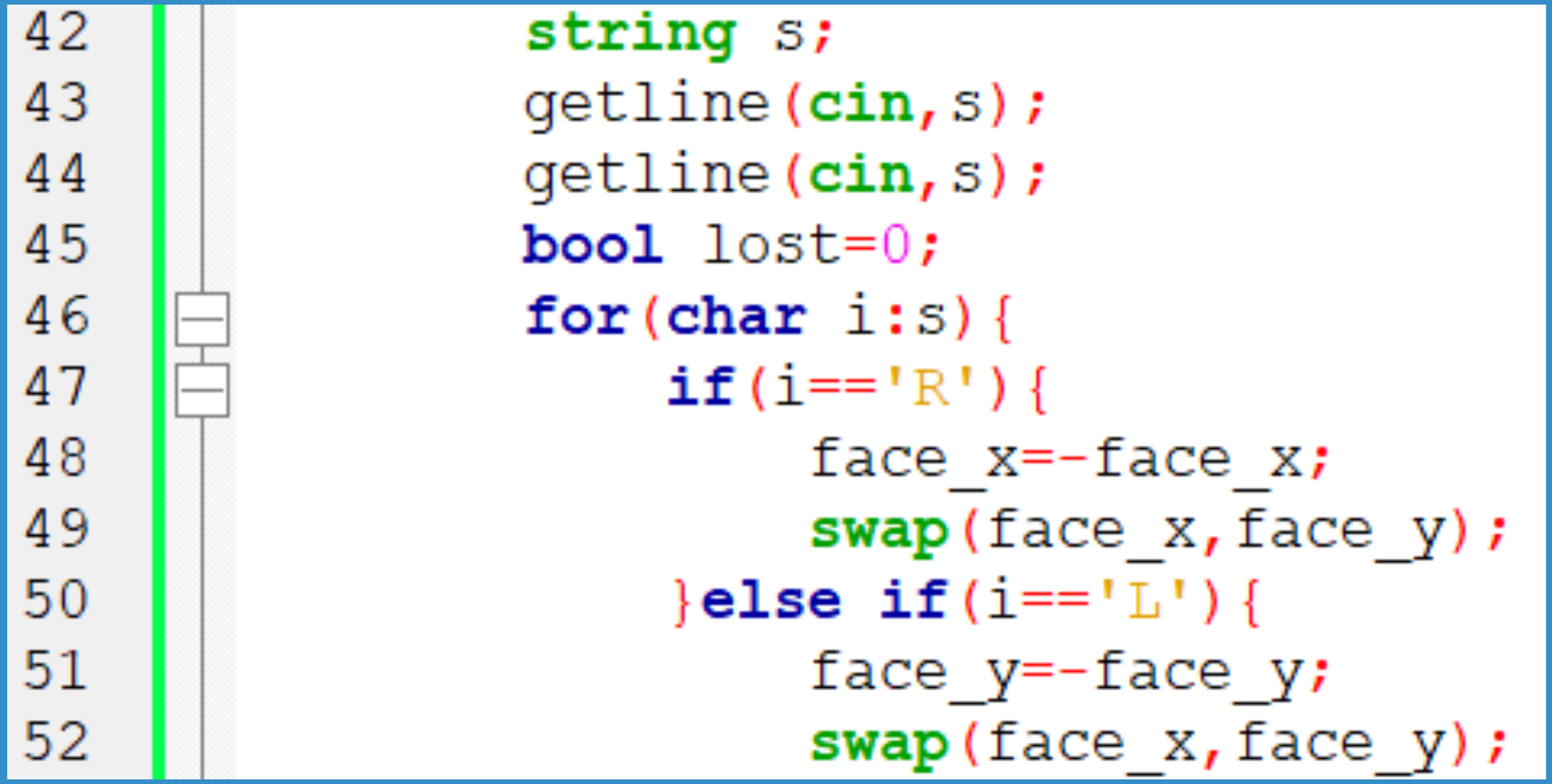


```
    int rux,ruy;
    cin>>rux>>ruy;
    vector<vector<bool>> vec(rux+1,vector<bool>(ruy+1,0));
    int x,y,face_x,face_y;
    char face;
    while(cin>>x>>y>>face){
        if(face=='E'){
            face_x=1;
            face_y=0;
        }else if(face=='W'){
            face_x=-1;
            face_y=0;
        }else if(face=='N'){
            face_x=0;
            face_y=1;
        }else if(face=='S'){
            face_x=0;
            face_y=-1;
        }
```

Step 1 ：輸入矩型大小跟初始位置

變數	註解
rux、ruy	矩形大小
x、y	初始位置
face、face_x、face_y	面向方向

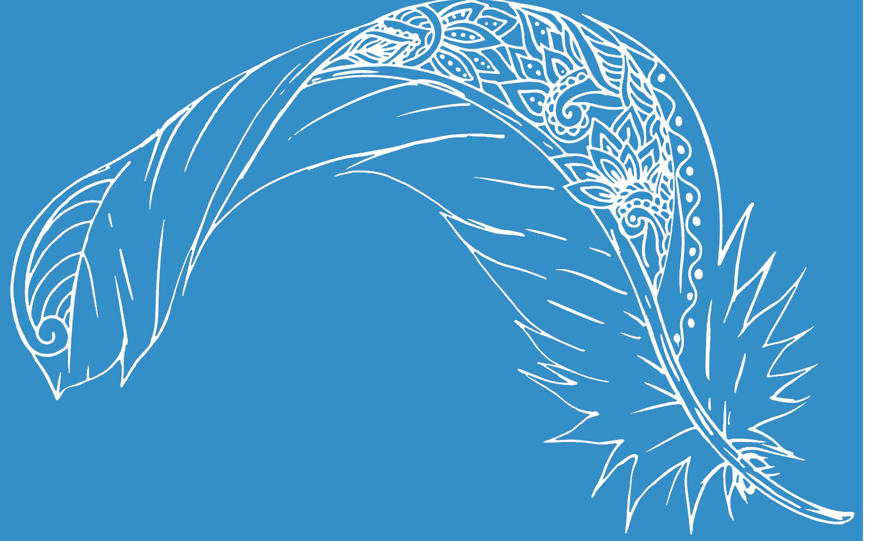
程式碼說明



```
        string s;
        getline(cin,s);
        getline(cin,s);
        bool lost=0;
        for(char i:s){
            if(i=='R'){
                face_x=-face_x;
                swap(face_x,face_y);
            }else if(i=='L'){
                face_y=-face_y;
                swap(face_x,face_y);
```

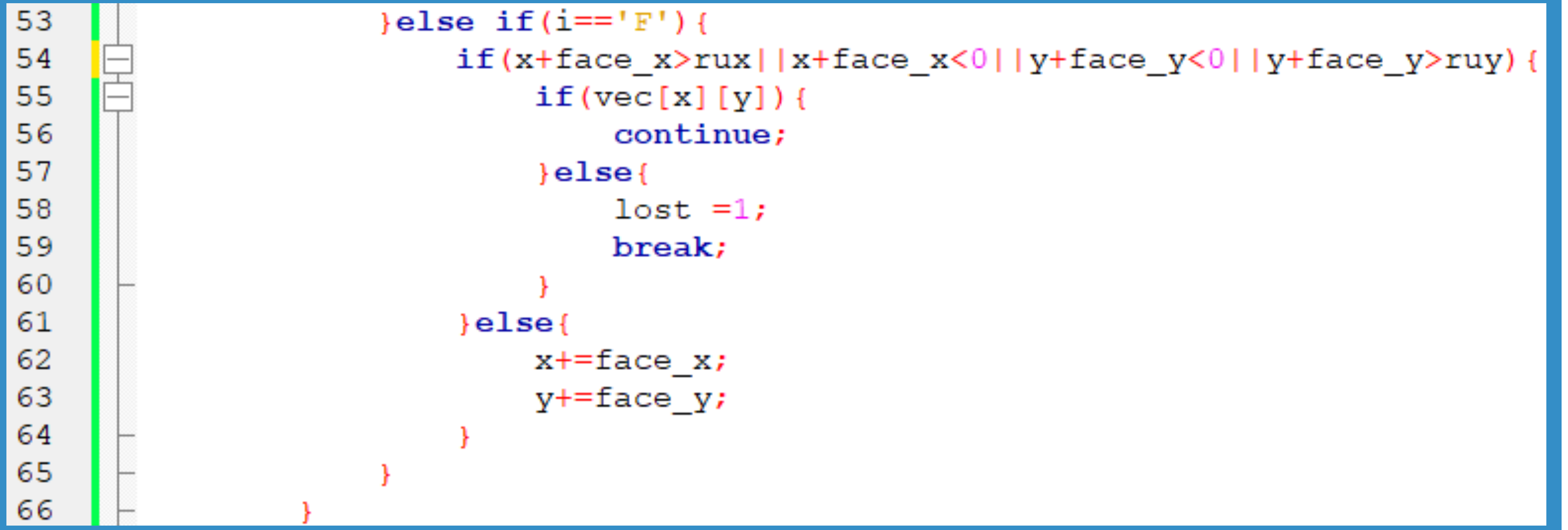
Step 2 ：輸入指令

變數	註解
x、y	初始位置
face、face_x、face_y	面向方向
s	指令



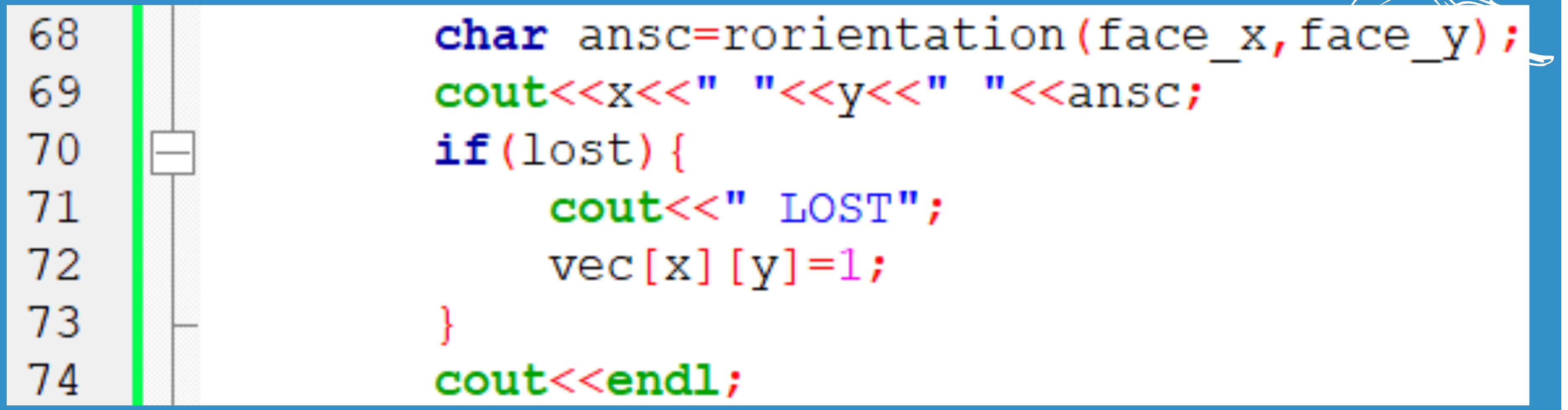
程式碼說明

Step 3 ：判斷有無超出邊界

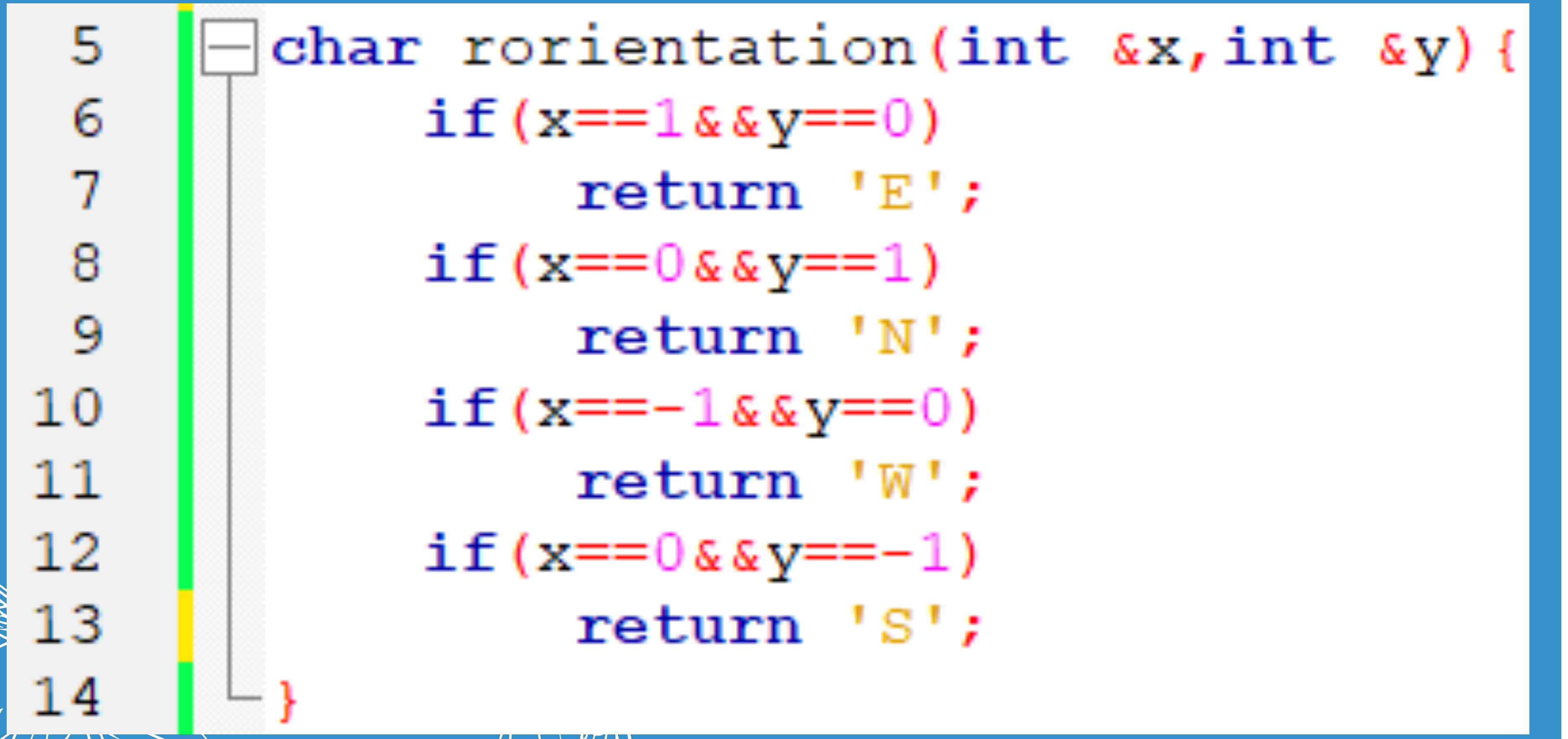


```
            }else if(i=='F'){
                if(x+face_x>rux||x+face_x<0||y+face_y<0||y+face_y>ruy){
                    if(vec[x][y]){
                        continue;
                    }else{
                        lost =1;
                        break;
                    }
                }else{
                    x+=face_x;
                    y+=face_y;
                }
            }
        }
```

變數	註解
x、y	初始位置
face、face_x、face_y	面向方向
vec	紀錄機器人有沒有掉下去過
s	指令



```
        char ansc=rorientation(face_x,face_y);
        cout<<x<<" "<<y<<" "<<ansc;
        if(lost){
            cout<<" LOST";
            vec[x][y]=1;
        }
        cout<<endl;
```



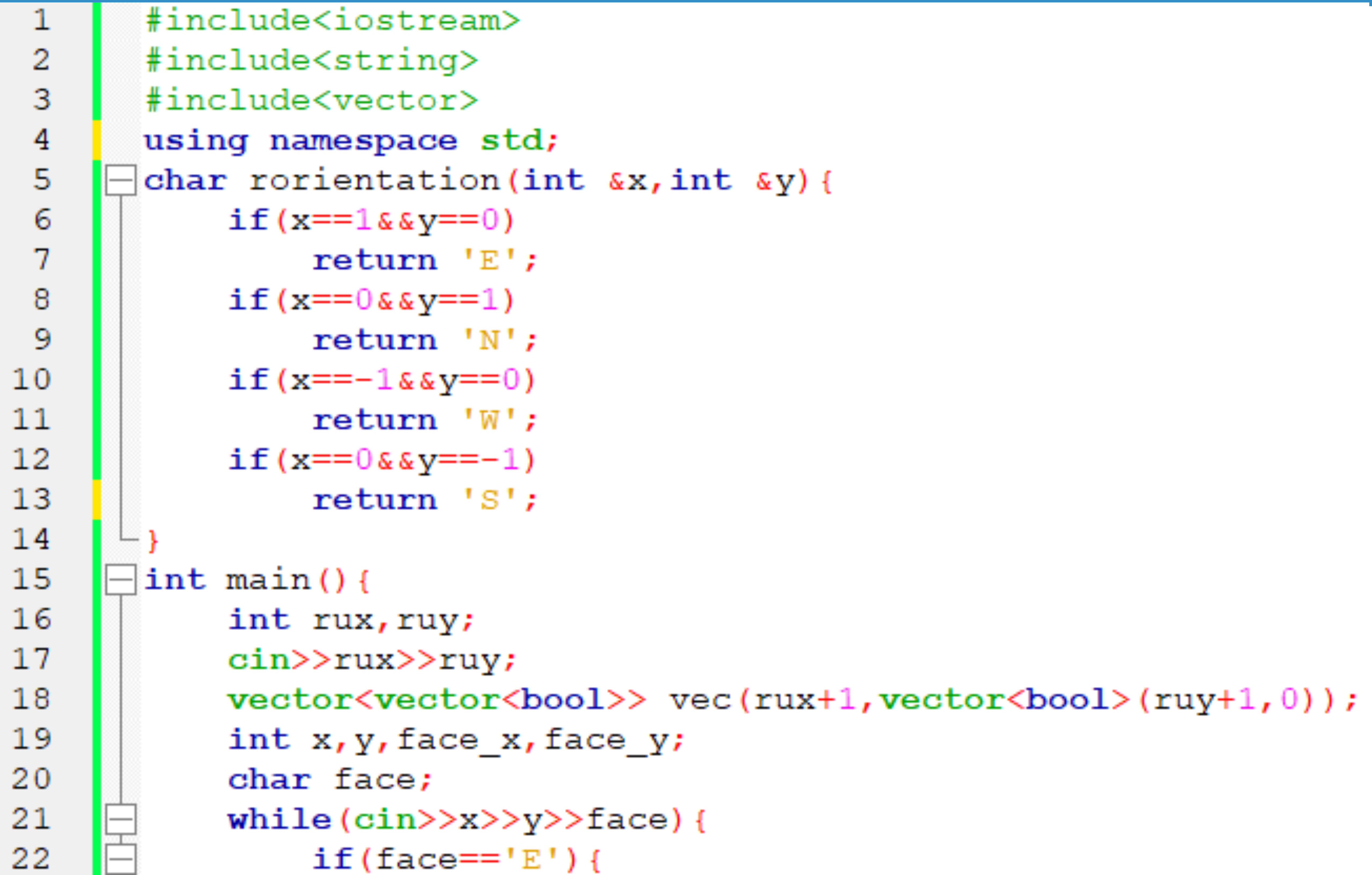
```
char rorientation(int &x,int &y){
    if(x==1&&y==0)
        return 'E';
    if(x==0&&y==1)
        return 'N';
    if(x==-1&&y==0)
        return 'W';
    if(x==0&&y==-1)
        return 'S';
}
```

程式碼說明

Step 4 ：輸出

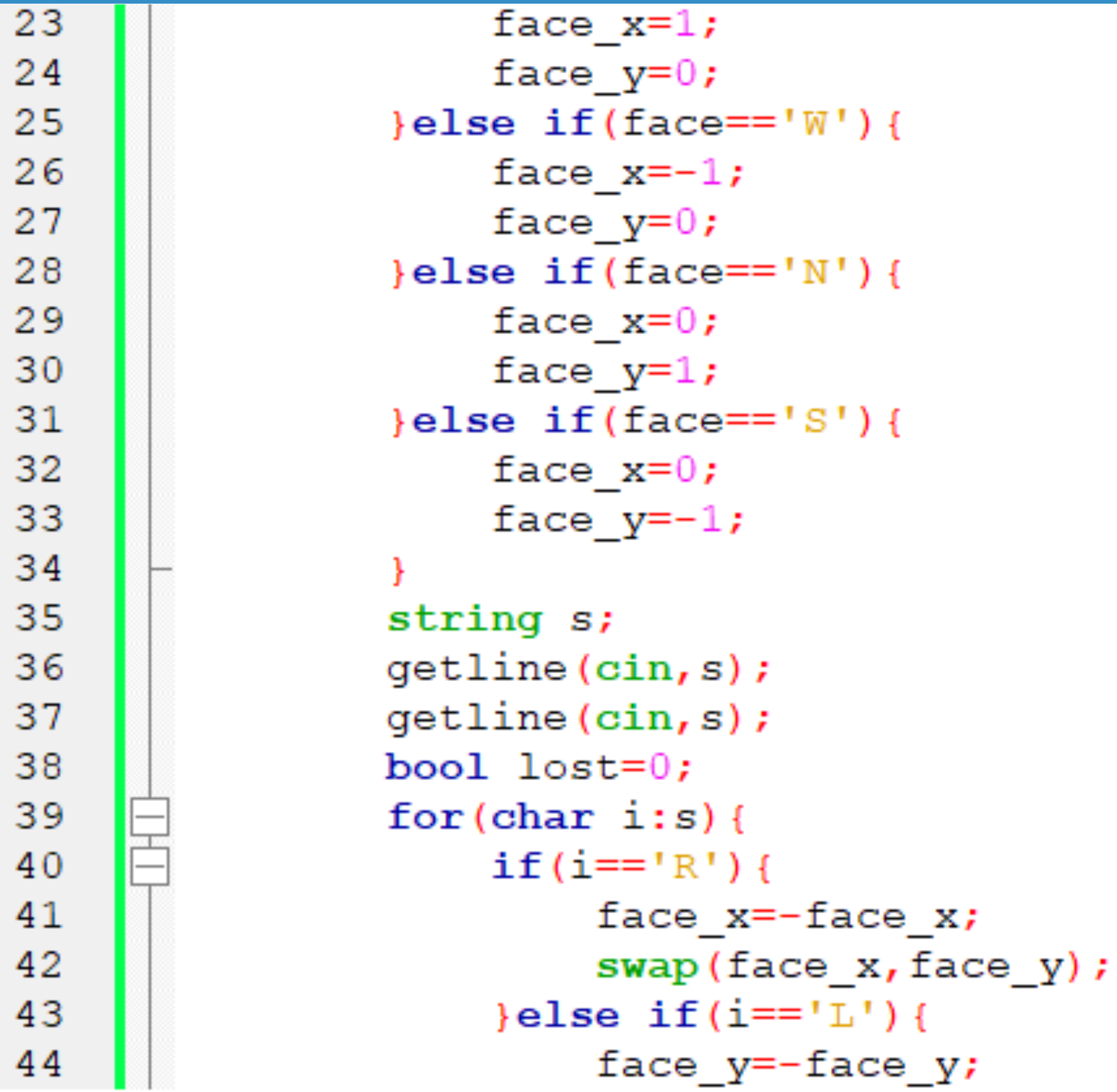
變數	註解
x、y	初始位置
face、face_x、face_y	面向方向
s	指令
ansc	最終面向方向

完整程式碼



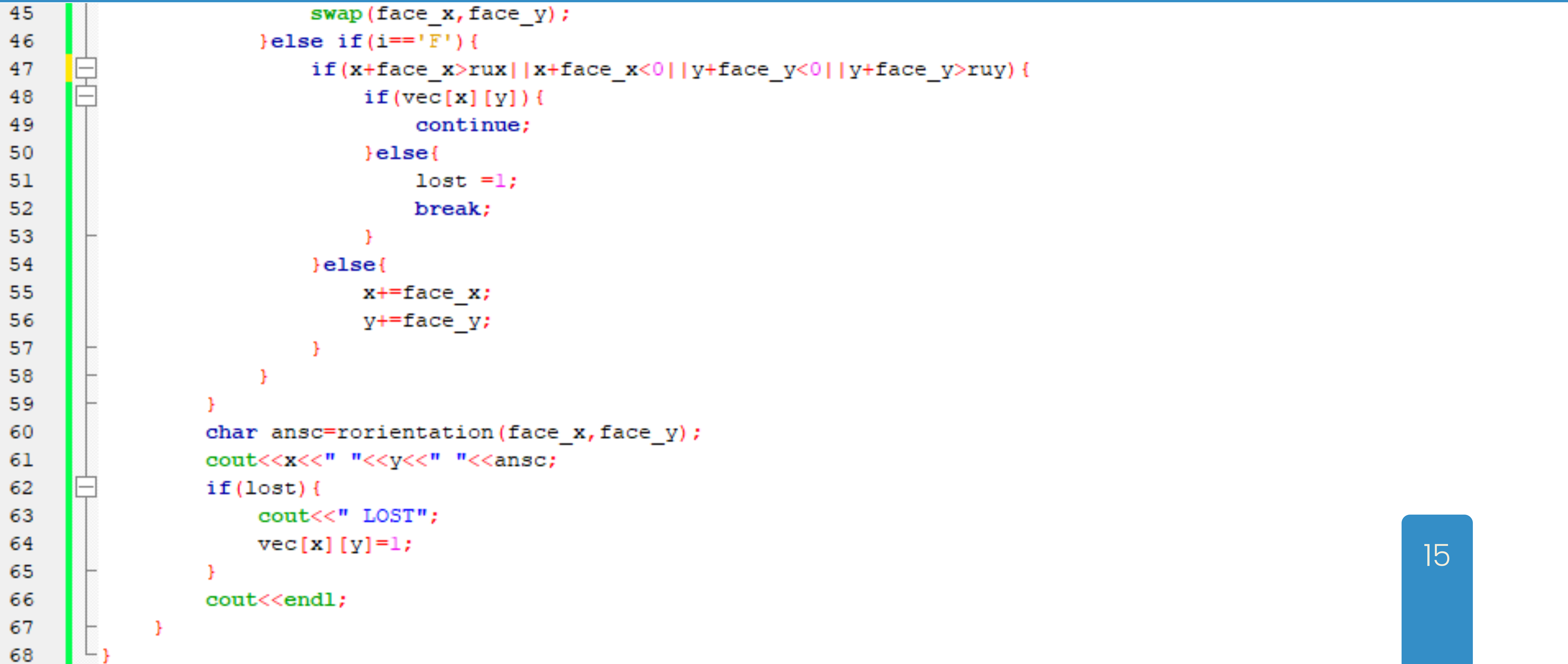
```
#include<iostream>
#include<string>
#include<vector>
using namespace std;
char rorientation(int &x,int &y){
    if(x==1&&y==0)
        return 'E';
    if(x==0&&y==1)
        return 'N';
    if(x==-1&&y==0)
        return 'W';
    if(x==0&&y==-1)
        return 'S';
}
int main(){
    int rux,ruy;
    cin>>rux>>ruy;
    vector<vector<bool>> vec(rux+1,vector<bool>(ruy+1,0));
    int x,y,face_x,face_y;
    char face;
    while(cin>>x>>y>>face){
        if(face=='E'){
```

完整程式碼



```
        face_x=1;
        face_y=0;
    }else if(face=='W'){
        face_x=-1;
        face_y=0;
    }else if(face=='N'){
        face_x=0;
        face_y=1;
    }else if(face=='S'){
        face_x=0;
        face_y=-1;
    }
    string s;
    getline(cin,s);
    getline(cin,s);
    bool lost=0;
    for(char i:s){
        if(i=='R'){
            face_x=-face_x;
            swap(face_x,face_y);
        }else if(i=='L'){
            face_y=-face_y;
```

完整程式碼



```
                    swap(face_x,face_y);
                }else if(i=='F'){
                    if(x+face_x>rux||x+face_x<0||y+face_y<0||y+face_y>ruy){
                        if(vec[x][y]){
                            continue;
                        }else{
                            lost =1;
                            break;
                        }
                    }else{
                        x+=face_x;
                        y+=face_y;
                    }
                }
            }
            char ansc=rorientation(face_x,face_y);
            cout<<x<<" "<<y<<" "<<ansc;
            if(lost){
                cout<<" LOST";
                vec[x][y]=1;
            }
            cout<<endl;
        }
}
```

資料來源

英文題目：
https://vjudge.net/problem/UVA-118

中文題目：
https://zerojudge.tw/ShowProblem?problemid=c082

Thank you for listening!