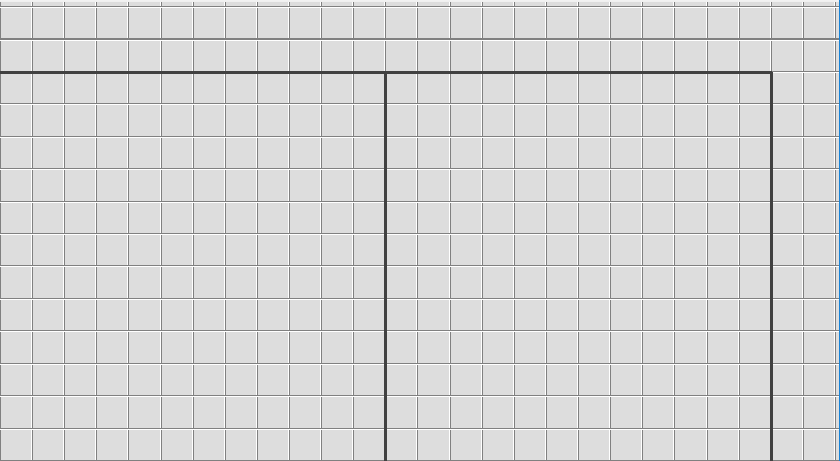
### Brief Description

I implemented a robot application that uses the Robot class. This class is also described in a document called “RobotWorld.pdf”’

The C++ program will create a robot in a location of which the x and y coordinates are inputs from the user. There are some limitations for these input x and y coordinates, which will be explained in a later section. Then, program will create a mirror robot in the symmetrical position with respect to the middle line (mirror) in the robot world. Then, the original robot will begin a clockwise spiral movement starting from going to south by 1 cell. After completing 2 cycles (making a total of 8 moves) it will stop. Mirror robot will follow original robot’s movements symmetrically in a counter-clockwise spiral movement starting from going to south by 1 cell. Just like the original robot, the mirror robot will also stop after completing 2 cycles (making a total of 8 moves). A screenshot of the empty world is given below:



### Details and Rules

First, the position of the original robot will be entered by the user. The coordinates of this robot must be non-negative integers between 0 and 11 inclusively (may be equal to 0 or 11). Otherwise, program should display an appropriate error message immediately and should not even prompt for the other inputs and of course **no** robot should be created.

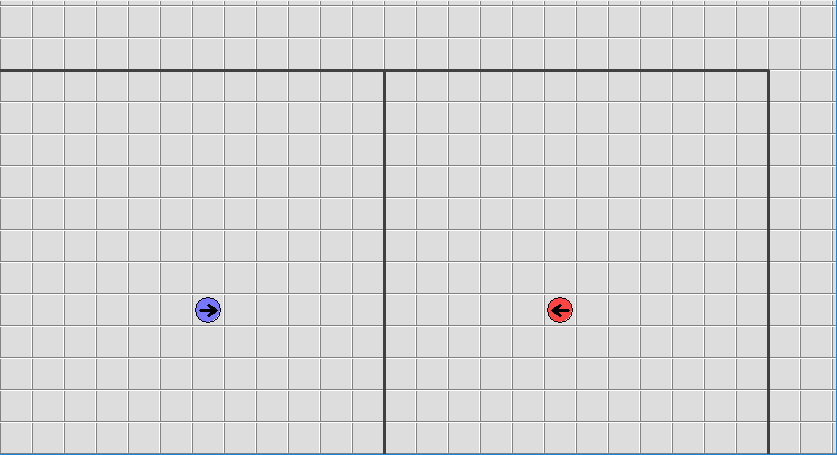
The world is a rectangle (24x12) and it is divided into two squares (12x12) with a line in the middle shown in the figure below. The mirror robot should be created in the right square and it should be symmetric of the original robot in the left square with respect to the dividing line in the middle of two squares.

The initial orientation (the direction that the robot faces when created) is NOT an input and it should be **east** for the original robot and **west** for the mirror robot.

Then program should set the colors of both robots; the color of the original robot must be set to

**blue**, and the color of the mirror robot must be set to **red**.

The world may look something like this when the robots are first created for input coordinates (6,4) entered for the original robot:



At this point, the original robot will begin its spiral movement starting from going to south by 1 cell. Spiral movement of the original robot should be in a clockwise fashion whereas the mirror robot in a counter-clockwise fashion. Which means after going to south, the original robot should turn to west whereas the mirror robot should turn to east before moving again. After moving to west or east by 1 cell, the robots then should turn to north. In order to make a proper spiral movement, the distance the original robot will travel should be increased to 2 at this point. After completing the first cycle (4 moves) we expect robots to change colors; the original robot should be **green** and the mirror robot should be **purple** from this point on. We expect the both robots to move a total of 8 times, the necessary distances to achieve spiral movement should be determined.

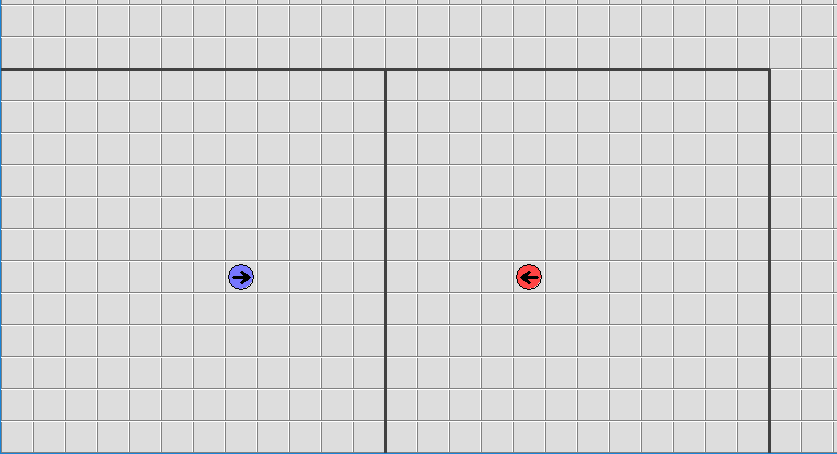
If robots’ movement is blocked by a wall during the spiral movement, they must not move into the wall, instead display an appropriate message and end the program. But keep in mind that, even though the robots must not move into the wall, they should move until they are next to the wall.

While the original robot makes its moves, the mirror robot will mimic its movements symmetrically. Which means; the mirror robot moves when the original robot moves, such that its movement is symmetric with respect to the mirror wall. Refer to the following table for movements:

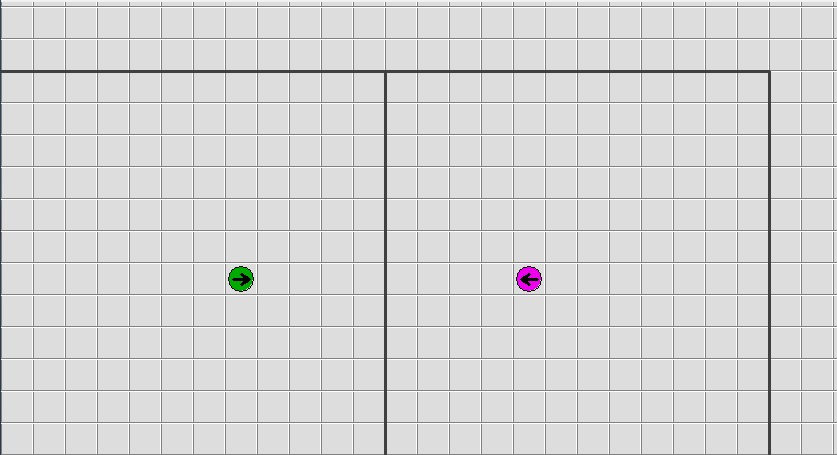
|  |  |
| --- | --- |
| **Original Robot** | **Mirror Robot** |
| East | West |
| West | East |
| North | North |
| South | South |

Keep in the mind that the robots cannot move at the same time. The original robot should move towards a direction first, and then the mirror robot should make its mimic movement before the original robot makes its next movement.

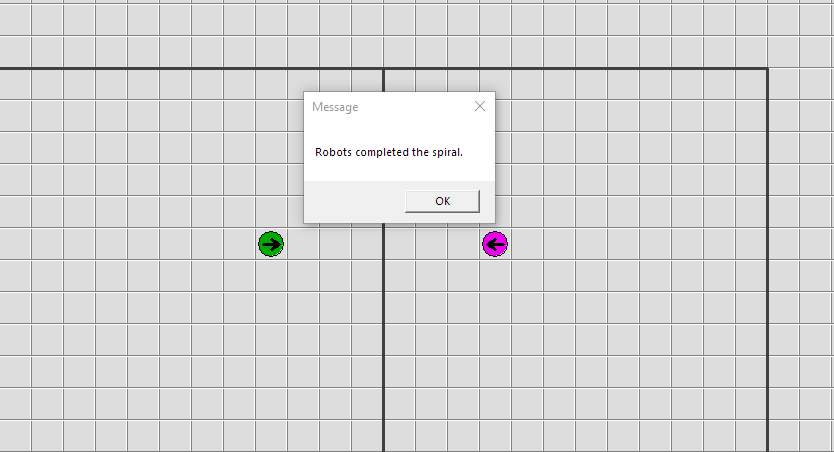
In the screenshot below, an example movement flow for the robot created at (6,4). After completing the first cycle, but before changing the colors:



After changing the colors:

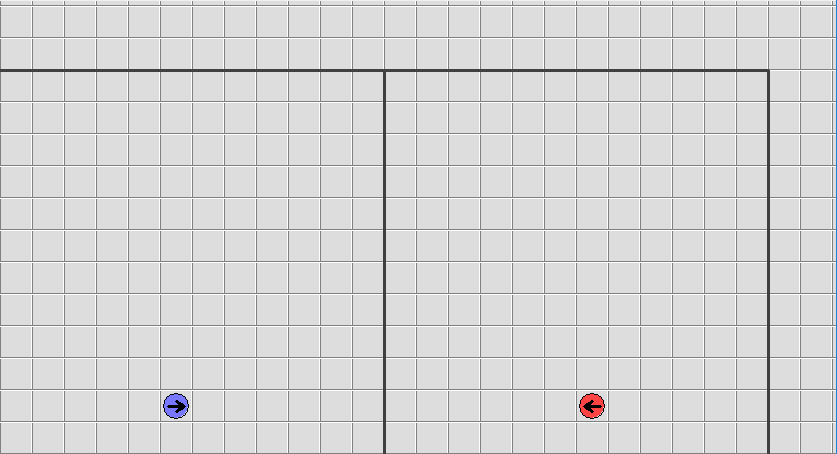


End of the spiral movement:

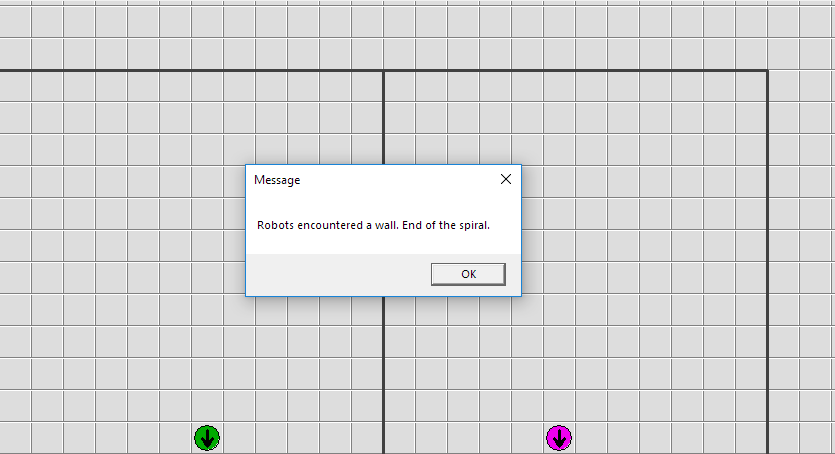


Below there is another example with the original robot created at (5,1), please notice that the spiral movement cannot be completed because of the wall but the robots do **not** die, instead an appropriate message is shown.

When robots are created:



The end result:



### The Program Flow

* Ask the user to input x coordinate of the original robot. Check the input value according to the following rule:
  + x-coordinate of the robot: 0 ≤ x < 12

If the value is correct, proceed, otherwise terminate the program. Inputs are assumed to be integers.

* Ask the user to input y coordinate of the original robot. Check the input value according to the following rule:
  + y-coordinate of the robot: 0 ≤ y < 12

If the value is correct, proceed, otherwise terminate the program. Inputs are assumed to be integers.

* Create the original robot at the given coordinates, facing east.
* Create the mirror robot at the symmetrical position according to the mirror, facing west.
* Set their colors; the original robot should be **blue**, the mirror robot should be **red**.
* The original robot will begin its spiral movement by going 1 cell to south. Remember that, the mirror robot should mimic its movements symmetrically with respect to the mirror line, after each move of the original robot.
* After completing a cycle (4 moves), robots should change colors; the original robot should be

**green** and the mirror robot should be **purple**.

* After completing two cycles (8 moves), robots should stop their movements.
* If robots’ movement is blocked by a wall during the spiral movement, they must not move into the wall, instead display an appropriate message saying “*Robots encountered a wall. End of spiral.*” and end the program. Even though the robots must not move into the wall, they should move until they are next to the wall.
* If robots complete the spiral movement, then program should display an appropriate message saying “*Robots completed the spiral.*” before ending.