**ROBOMAZE**

**ABSTRACT:**

ROBOMAZE is a puzzle based project. Here we observe the movement of robo in the maze. The maze has corridors and walls, and robot can move only in one of the four different directions: North, South, East, and West. The maze will be given in the input, with a grid of characters `X' and `.', where `X' denotes a wall and `.' denotes an empty space. The robot can move through empty spaces, but not through the wall. You can also assume that the robot cannot leave the maze; which is equivalent to the whole maze being surrounded by a wall. After the maze you will be given the initial position of the robot with two numbers specifying row and column (the top left position is (1,1)). You will assume that the robot is facing North (up on the maze grid). Following this, you will be given a sequence of commands consisting of letters, with possibly having whitespace between them. The letter commands are as follows: R is a command to rotate the robot 90 degrees clockwise (to the right), L is a command to rotate the robot 90 degrees counter-clockwise (to the left), F moves the robot forward one cell, unless there is a wall preventing this, in which case the robot does nothing, and Q denotes the final position. You should print out the current robot row, column, and orientation, and assume that the next test case follows. You will also print the original map with the letter `R' being placed on all squares visited by the robot.

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **TITLE** | | **PAGE NO** |
| 1 | Introduction | | 1 |
| 2 | Aim of the Project | | 2 |
| 2.1 | Advantages & Disadvantages | | 2 |
| 3 | Software & Hardware Details | | 3 |
| 4 | Algorithm | | 4 |
| 5 | Implementation | | 6 |
| 6 | outputs | | 10 |
| 7 | | Conclusion | 11 |
|  | |  |  |
|  |  | |  |

**INTRODUCTION**

In this project ”ROBOMAZE” which accepts the maze pattern , command , initial position and display the end point of the robo. Here we will start the program by showing an input screen to the user where we will accept the no of test cases , pattern ,and command .

**Header file:**

Here the header files used are **stdio.h** and **conio.h** these are predefined functions . stdio.h has predefined functions of printf and scanf functions , conio.h is used to provide console input/output .

**Declaration:**

The data structures declared in this program are **int** and **char** as we use numbers and characters in this program . No other data type is necessary to use .

* The array is used to know the position of the robo with respect to no of rows and no of columns .
* For loop is used to ask the input as per the given command the program executes as follows .
* The program will print the final row, column, and orientation of the robot for each test case, followed by the map of robot movements. The orientation is denoted as N (for North or up), S (for South or down), E (for East or left), and W (for West or right). The map that follows will have the letter `R' placed in all squares visited by the robot.

**AIM**

**the program is to simulate the robo movement in the maze**

**Advantages:-**

* The code is used to move the robo from initial position to the final point
* The robo can be moved in such a way that it can .

**Disadvantages:-**

* The robo can only move through the empty space denoted by ‘.’ And it can’t move through the wall .

**SYSTEM REQUIREMENTS**

* **SOFTWARE REQUIREMENTS:**

The major software requirements of the project are as follows:

Language : Turbo-C

Operating system **:** Windows Xp or later.

* **HARDWARE REQUIREMENTS:**

The hardware requirements that map towards the software are as follows:

RAM : above 2GB

Processor : INTEL CORE I5 8TH GEN

**ALGORITHM**

**1**. Start

**2.** Read orient=N, comn, rows, columns , x , y , i , j ;

Read pattern;

**2.1** if(comn[i]==’R’&&orient==’N’)

Orient=’E’;

**2.2** else if(comn[i]==’R’&&orient==’E’)

Orient=’S’;

**2.3** else if(comn[i]==’R’&&orient==’S’)

Orient=’W’;

**2.4** else if(comn[i]==’R’&&orient==’W’)

Orient=’N’;

**3.** if(comn[i]==’L’&&orient==’N’)

Orient=’W’;

**3.1** else if(comn[i]==’L’&&orient==’E’)

Orient=’N’;

**3.2** else if(comn[i]==’L’&&orient==’S’)

Orient=’E’;

**3.3** else if(comn[i]==’L’&&orient==’W’)

Orient=’S’;

**4.** if(orient==’N’&&comn[i]==’F’&&a[x-1][y]!=’x’)

x=x-1;a[x][y]=’R’;

**4.1** else if(orient==’E’&&comn[i]==’F’&&a[x][y+1]!=’x’)

y=y+1;a[x][y]=’R’;

**4.2** else if(orient==’N’&&comn[i]==’F’&&a[x+1][y]!=’x’)

x=x+1;a[x][y]=’R’;

**4.3** else if(orient==’W’&&comn[i]==’F’&&a[x][y-1]!=’x’)

y=y-1;a[x][y]=’R’;

**5.** else

comn[i]==’Q’;

x,++y,orient;

**6.** for every i in range 1 to n

for every j in range 1 to n

**7.** print a[i][j];

**8.** stop

**IMPLEMENTATION**

#include<stdio.h>

#include<conio.h>

void main()

{

char a[10][10],orient='N',comn[100];

int m,n,x,y,i,j;

clrscr();

printf("enter rows and columns");

scanf("%d%d",&m,&n);

printf("Enter the maze pattern\n");

for(i=1;i<=m;i++)

scanf(" %[^\n]s",a[i]);

printf("enter robo position");

scanf("%d %d",&x,&y);

a[x][--y]='R';

printf("\nEnter the command:");

scanf(" %[^\n]s",comn);

for(i=0;comn[i]!='\0';i++)

{

if(comn[i]=='R' && orient=='N')

orient='E';

else if(comn[i]=='R' && orient=='E')

orient='S';

else if (comn[i]=='R' && orient =='S')

orient='W';

else if(comn[i]=='R' && orient=='W')

orient='N';

if(comn[i]=='L' && orient=='N')

orient='W';

else if(comn[i]=='L' && orient=='E')

orient='N';

else if (comn[i]=='L' && orient =='S')

orient='E';

else if(comn[i]=='L' && orient=='W')

orient='S';

if (orient=='N' && comn[i]=='F' && a[x-1][y]!='X')

{

x=x-1;

a[x][y]='R';

}

else if (orient=='E' && comn[i]=='F' && a[x][y+1]!='X')

{

y=y+1;

a[x][y]='R';

}

else if (orient=='S' && comn[i]=='F' && a[x+1][y]!='X')

{

x=x+1;

a[x][y]='R';

}

else if (orient=='W' && comn[i]=='F' && a[x][y-1]!='X')

{

y=y-1;

a[x][y]='R';

}

//Add code here

else if(comn[i]=='Q')

break;

}//for loop closes here

printf("%d %d %c\n",x,++y,orient);

for(i=1;i<=m;i++)

{

for(j=0;j<n;j++)

printf("%c",a[i][j]);

printf("\n");

}

}

**OUTPUTS**

**Screen Shots:**

Test case 01 :



**Test case 02 : **

**CONCLUSION**

Thus , we have find the final position and the direction of the robot . Finally it is concluded that the based on the orientation, commands, and the empty spaces the robot reaches its final position with also the direction can be known and displayed.