Quickgrid

2D array adjacent square check using coordinate directions

UVA Problem 10189 – Minesweeper Solution

Posted on **October 29, 2015October 29, 2015** by **quickgrid**
UVA Problem 10189 – Minesweeper Solution:

Click here to go to this problem in uva Online Judge. (https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=24&page=show_problem&problem=1130)

Solving Technique:

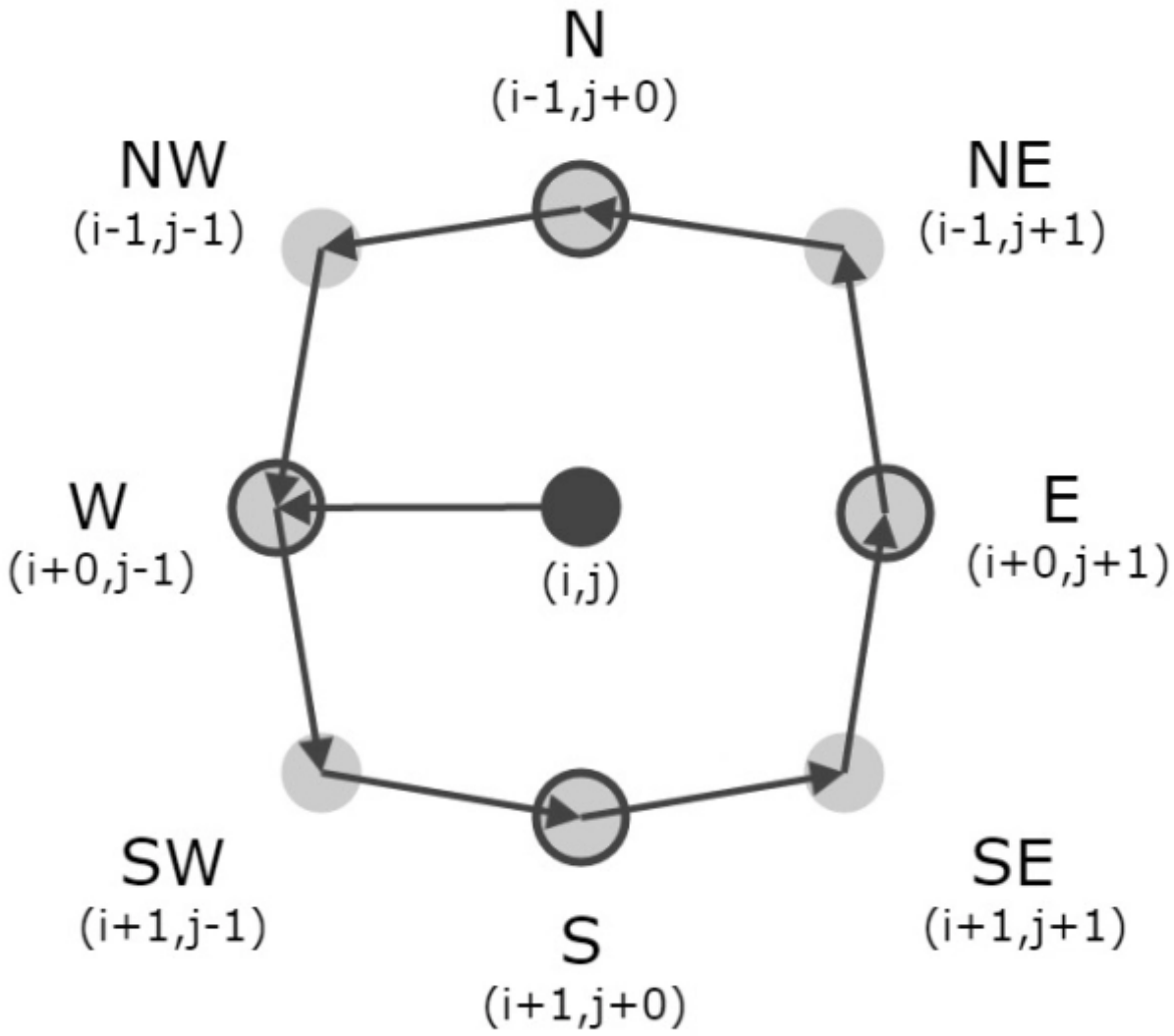
Given a mine field that is a matrix / 2D array, produce an output that contains count of adjacent mines for each squares.

In a 2D array for each squares there are at most adjacent 8 squares. If the current position is i and j then,

$$\begin{bmatrix} (i-1,j-1) & (i-1,j) & (i-1,j+1) \\ (i+0,j-1) & (i,j) & (i+0,j+1) \\ (i+1,j-1) & (i-1,j) & (i+1,j+1) \end{bmatrix}$$

Just traverse the matrix row-column wise and check its adjacent squares for getting mine count for current position. The adjacent squares check can be implemented with 8 if conditions for each one.

Another technique is to store the co-ordinates of adjacent squares and using a for loop to check them. The illustration below shows how the 2nd code is implemented using arrays and for loops,



(https://quickgrid.files.wordpress.com/2015/10/uva-10189-matrix-adjacent-squares-traverse.jpg)

uva 10189 matrix adjacent squares check with saved co-ordinates and for loop

Here the arrow from the center shows where checking starts. The 1D arrays drow and dcol hold the row and column values the way shown above. It can be changed by modifying drow and dcol arrays.

Important: Be sure to add or print a new line after each output unless otherwise specified. The outputs should match exactly because sometimes even a space character causes the answer to be marked as wrong answer. Please compile with c++ compiler as some of my codes are in c and some in c++.

More Inputs of This Problem on uDebug. (http://www.udebug.com/UVa/10189)

Input:

```
4 4
*...
....
.*..
....
3 5
**...
.....
.*...
0 0
```

Output:

```
Field #1:
*100
2210
1*10
1110

Field #2:
**100
33200
1*100
```

Code Using Multiple if Conditions:

```
1  /**
2   * Author:    Asif Ahmed
3   * Site:      https://quickgrid.wordpress.com (https://quickgrid.wordpress.com)
4   * Problem:   UVA 10189 - Minesweeper
5   * Technique: 2D Array / Matrix Boundary checking using
6   *            if conditions.
7   */
8
9  #include<stdio.h>
10 #include<string.h>
11
12
13 #define MAXSIZE 101
14
15
16 static char MineField[MAXSIZE][MAXSIZE];
17
18
```

```
19
20  int main(){
21
22      //freopen("input.txt", "r", stdin);
23      //freopen("output.txt", "w", stdout);
24
25
26      int n, m;
27
28      int FieldNumber = 0;
29
30      while( scanf("%d%d", &n, &m), n ){
31
32          getchar();
33
34          for(int i = 0; i < n; ++i)
35              scanf("%s", &MineField[i]);
36
37
38          if( FieldNumber )
39              printf("\n");
40
41
42          for(int i = 0; i < n; ++i){
43              for(int j = 0; j < m; ++j){
44
45                  if( MineField[i][j] == '*' )
46                      continue;
47
48                  int temp = 0;
49
50                  if( i + 1 < n && MineField[i + 1][j] == '*' )
51                      ++temp;
52                  if( i + 1 < n && j + 1 < m && MineField[i + 1][j + 1] == '*' )
53                      ++temp;
54                  if( j + 1 < m && MineField[i][j + 1] == '*' )
55                      ++temp;
56                  if( i - 1 >= 0 && j + 1 < m && MineField[i - 1][j + 1] == '*' )
57                      ++temp;
58                  if( i - 1 >= 0 && MineField[i - 1][j] == '*' )
59                      ++temp;
60                  if( i - 1 >= 0 && j - 1 >= 0 && MineField[i - 1][j - 1] == '*' )
```

```
                    ++temp;
                if( j - 1 >= 0 && MineField[i][j - 1] == '*' )
                    ++temp;
                if( i + 1 < n && j - 1 >= 0 && MineField[i + 1][j - 1] == '*' )
                    ++temp;


            MineField[i][j] = temp + '0';

            }
         }


         printf("Field #%d:\n", ++FieldNumber);


        for(int i = 0; i < n; ++i){
            for(int j = 0; j < m; ++j)
                putchar(MineField[i][j]);
            printf("\n");
        }

    }

    return 0;
}
```

Code Bound checking using Array & for Loop:

```
/**
 * Author:    Asif Ahmed
 * Site:      https://quickgrid.wordpress.com (https://quickgrid.wordpress.com)
 * Problem:   UVA 10189 - Minesweeper
 * Technique: 2D Array / Matrix Boundary checking using
 *            co-ordinate array and for loop.
 */
```

```
 8
 9  #include<stdio.h>
10  #include<string.h>
11
12
13  #define MAXSIZE 101
14
15
16  static char MineField[MAXSIZE][MAXSIZE];
17
18
19  // Co-ordinates / directions of adjacent 8 squares.
20  // W, SW, S, SE, E, NE, N, NW
21  static const int drow[] = {0, 1, 1, 1, 0, -1, -1, -1};
22  static const int dcol[] = {-1, -1, 0, 1, 1, 1, 0, -1};
23
24
25
26  int main(){
27
28      //freopen("input.txt", "r", stdin);
29      //freopen("output.txt", "w", stdout);
30
31
32      int n, m;
33
34      int FieldNumber = 0;
35
36      while( scanf("%d%d", &n, &m), n ){
37
38          getchar();
39
40          for(int i = 0; i < n; ++i)
41              scanf("%s", &MineField[i]);
42
43
44          if( FieldNumber )
45              printf("\n");
46
47
48          for(int i = 0; i < n; ++i){
49              for(int j = 0; j < m; ++j){
```

```
50
51                  int temp = 0;
52
53                  // If mine found do nothing.
54                  if( MineField[i][j] == '*' )
55                      continue;
56
57
58                  // For each adjacent squares of the current square calculate mine count.
59                  // and set the count in current square.
60                  for(int k = 0; k < 8; ++k){
61
62                      // Check if out of bound of the 2D array or matrix.
63                      if( i + drow[k] < 0 || j + dcol[k] < 0 || i + drow[k] >= n || j + dcol[k] >= m )
64                          continue;
65
66                      // Check the appropriate co-ordinate for mine, if mine found increase count.
67                      if( MineField[i + drow[k] ][j + dcol[k]] == '*' )
68                          ++temp;
69
70                  }
71
72                  // All adjacent squares checked set the mine count for current squares.
73                  MineField[i][j] = temp + '0';
74
75              }
76          }
77
78
79          printf("Field #%d:\n", ++FieldNumber);
80
81
82         for(int i = 0; i < n; ++i){
83              for(int j = 0; j < m; ++j)
84                  putchar(MineField[i][j]);
85              printf("\n");
86         }
87
88      }
89
90      return 0;
91  }
```

Posted in **UVA** Tagged **2D array adjacent square check using coordinate directions**, **2d array boundary check**, **acm**, **c**, **code**, **competitive**, **contest**, **cpp**, **explain**, **explanation**, **matrix adjacent square check using for loop**, **matrix array boundary check**, **Online Judge**, **programming**, **solution**, **solve**, **technique**, **tutorial**, **UVA**, **uva 10189**, **uva 10189 minesweeper**, **very easy** **Leave a comment**