```
File - D:\cpl\2023-cpl-coding-0\5-function\game-of-life.c
 1 //
 2 // Created by hfwei on 2023/10/19.
 3 //
 4
 5 #include <stdio.h>
 6 #include <stdlib.h>
 7 #include <unistd.h>
 8 #include <synchapi.h>
10 #define SIZE 6
11 const int board[SIZE][SIZE] = {
        { 0 },
        { 0, 1, 1, 0, 0, 0 },
13
14
        { 0, 1, 1, 0, 0, 0 },
15
        { 0, 0, 0, 1, 1, 0 },
16
        { 0, 0, 0, 1, 1, 0 },
17
        { 0 }
18 };
19
20 //const int board[SIZE][SIZE] = {
21 //
          [1][1] = 1, [1][2] = 1,
          [2][1] = 1, [2][2] = 1,
22 //
23 //
          [3][3] = 1, [3][4] = 1,
24 //
          [4][3] = 1, [4][4] = 1
25 //};
26
27 int main() {
     // extended board
     int old_board[SIZE + 2][SIZE + 2] = { 0 };
29
30
31
     for (int row = 1; row <= SIZE; row++) {</pre>
32
        for (int col = 1; col <= SIZE; col++) {</pre>
33
          old_board[row][col] = board[row - 1][col - 1];
34
        }
35
     }
36
37
     // print the original board
     for (int row = 1; row <= SIZE; row++) {</pre>
38
39
        for (int col = 1; col <= SIZE; col++) {</pre>
          printf("%c ", old_board[row][col] ? '*' : ' ');
40
        }
41
42
        printf("\n");
43
44
     system("clear"); // clear the screen/terminal
45
     int new_board[SIZE + 2][SIZE + 2] = { 0 };
46
47
48
     for (int round = 1; round < 10; round++) {</pre>
        for (int row = 1; row <= SIZE; row++) {</pre>
49
50
          for (int col = 1; col <= SIZE; col++) {</pre>
51
            // count the number of neighbours of old_board[row][col]
52
            int neighbours =
                old_board[row - 1][col - 1] +
53
```

```
File - D:\cpl\2023-cpl-coding-0\5-function\game-of-life.c
                      old_board[row - 1][col] +
 54
 55
                      old_board[row - 1][col + 1] +
 56
                      old_board[row][col - 1] +
 57
                      old_board[row][col + 1] +
                      old_board[row + 1][col - 1] +
 58
 59
                      old_board[row + 1][col] +
                      old_board[row + 1][col + 1];
 60
 61
 62
             // evaluate the new board
             if (old_board[row][col]) { // old_board[row][col] is alive
 63
               new_board[row][col] = (neighbours == 2 || neighbours == 3);
 64
             } else { // old_board[row][col] is dead
 65
               new_board[row][col] = (neighbours == 3);
 66
             }
 67
           }
 68
         }
 69
 70
 71
         // print the new board
 72
         for (int row = 1; row <= SIZE; row++) {</pre>
 73
           for (int col = 1; col <= SIZE; col++) {</pre>
 74
             printf("%c ", new_board[row][col] ? '*' : ' ');
 75
 76
           printf("\n");
 77
         }
 78
 79
         // sleep for a while
         // Linux: #include <unistd.h>
 80
 81
         sleep(1);
         // Windows: #include <windows.h>: Sleep(ms)
 82
 83
         // Sleep(1000);
 84
 85
         // clear the screen
         // Linux: #include <stdlib.h>
 86
         system("clear");
 87
         // Windows: #include <stdlib.h> system("clr);
 88
 89
         // system("clr");
 90
 91
         // start the next round
 92
         for (int row = 1; row <= SIZE; row++) {</pre>
 93
           for (int col = 1; col <= SIZE; col++) {</pre>
 94
             old_board[row][col] = new_board[row][col];
           }
 95
 96
         }
       }
 97
 98
 99
      return 0;
100 }
```