

```
1 //
2 // Created by hengxin on 10/19/22.
3 // Run it with "Terminal"
4 //
5
6 #include <stdio.h>
7 #include <stdlib.h>
8 #include <unistd.h>
9
10 #define SIZE 6
11
12 // extended_board as a parameter
13 // 1D array: (int, the address of the first element), the length
14 // 2D array: array of arrays [1][3]:
15 // address + 1 * (sizeof int) * size of col + (sizeof int) * 3
16 // 0: -----
17 // 1: -----
18 // 2: -----
19 void ExtendBoard(const int origin_board[][SIZE],
20                 int extended_board[][SIZE + 2]);
21 void PrintExtendedBoard(const int extended_board[][SIZE + 2]);
22 void GenerateNewBoard(const int old_board[][SIZE + 2],
23                      int new_board[][SIZE + 2]);
24 void CopyExtendedBoard(const int src_board[][SIZE + 2],
25                       int dest_board[][SIZE + 2]);
26 void SleepAndClear(int sec);
27
28 int main() {
29     const int board[SIZE][SIZE] = {
30         { 0 },
31         { 0, 1, 1, 0, 0, 0 },
32         { 0, 1, 1, 0, 0, 0 },
33         { 0, 0, 0, 1, 1, 0 },
34         { 0, 0, 0, 1, 1, 0 },
35         { 0 }
36     };
37
38     int old_board[SIZE + 2][SIZE + 2] = { 0 };
39     ExtendBoard(board, old_board);
40     PrintExtendedBoard(old_board);
41     // SleepAndClear(1);
42
43     int new_board[SIZE + 2][SIZE + 2] = { 0 };
44     for (int round = 0; round < 10; round++) {
45         GenerateNewBoard(old_board, new_board);
46         SleepAndClear(1);
47         PrintExtendedBoard(new_board);
48         CopyExtendedBoard(new_board, old_board);
49     }
50
51     return 0;
52 }
53
```

```

54 void ExtendBoard(const int origin_board[][SIZE],
55                 int extended_board[][SIZE + 2]) {
56     for (int row = 0; row < SIZE + 2; row++) {
57         for (int col = 0; col < SIZE + 2; col++) {
58             if (row == 0 || row == SIZE + 1 || col == 0 || col == SIZE + 1
59 ) {
60                 extended_board[row][col] = 0;
61             } else {
62                 extended_board[row][col] = origin_board[row - 1][col - 1];
63             }
64         }
65     }
66
67 void PrintExtendedBoard(const int extended_board[][SIZE + 2]) {
68     for (int row = 1; row <= SIZE; row++) {
69         for (int col = 1; col <= SIZE; col++) {
70             printf("%c ", extended_board[row][col] ? '*' : ' ');
71         }
72         printf("\n");
73     }
74 }
75
76 void GenerateNewBoard(const int old_board[][SIZE + 2],
77                      int new_board[][SIZE + 2]) {
78     for (int row = 1; row <= SIZE; row++) {
79         for (int col = 1; col <= SIZE; col++) {
80             // count the number of neighbours of old_board[row][col]
81             int neighbours =
82                 old_board[row - 1][col - 1] +
83                 old_board[row - 1][col] +
84                 old_board[row - 1][col + 1] +
85                 old_board[row][col - 1] +
86                 old_board[row][col + 1] +
87                 old_board[row + 1][col - 1] +
88                 old_board[row + 1][col] +
89                 old_board[row + 1][col + 1];
90
91             // evaluate the new board
92             if (old_board[row][col]) { // old_board[row][col] is alive
93                 new_board[row][col] = (neighbours == 2 || neighbours == 3);
94             } else { // old_board[row][col] is dead
95                 new_board[row][col] = (neighbours == 3);
96             }
97         }
98     }
99 }
100
101 void CopyExtendedBoard(const int src_board[][SIZE + 2],
102                      int dest_board[][SIZE + 2]) {
103     for (int row = 1; row <= SIZE; row++) {
104         for (int col = 1; col <= SIZE; col++) {
105             dest_board[row][col] = src_board[row][col];

```

File - D:\cpl\2023-cpl-coding-0\5-function\game-of-life-transformed.c

```
106     }  
107 }  
108 }  
109  
110 void SleepAndClear(int sec) {  
111     sleep(sec);  
112     system("clear");  
113 }
```