

```
1 // Created by hfwei on 2024/10/16.
2
3 #include <stdio.h>
4
5 #define LEN_L 5
6 #define LEN_R 6
7
8 int L[LEN_L] = { 1, 3, 5, 7, 9 };
9 int R[LEN_R] = { 0, 2, 4, 6, 8, 10 };
10
11 int main(void) {
12     // TODO: merge L and R into a sorted array
13     int l = 0;
14     int r = 0;
15
16     while (l < LEN_L && r < LEN_R) {
17         if (L[l] <= R[r]) {
18             printf("%d ", L[l]);
19             l++;
20         } else {
21             printf("%d ", R[r]);
22             r++;
23         }
24     }
25
26     // l >= LEN_L || r >= LEN_R
27     while (r < LEN_R) {
28         printf("%d ", R[r]);
29         r++;
30     }
31
32     while (l < LEN_L) {
33         printf("%d ", L[l]);
34         l++;
35     }
36
37     return 0;
38 }
```

```
1 # 4-loops
2
3 - `Alt + 6`: Problems on the status bar
4 - `SonarLint` on the status bar
5
6 ## `game-of-life.c`
7
8 - play with it
9   - [wiki](https://en.wikipedia.org/wiki/Conway%
10     27s_Game_of_Life)
11   - [Demo](https://playgameoflife.com/)
12   - [Gosper_glider_gun](https://playgameoflife.com/lexicon
13     /Gosper_glider_gun)
14   - [LifeWiki](https://conwaylife.com/wiki/Main_Page)
15   - [Life Lexicon Home Page](https://conwaylife.com/ref/
16     lexicon/lex_home.htm)
17 - 2D-array
18   - initialization (Section 8.2.1)
19     - row-major
20     - row by row
21     - indicator
22 - extension of board
23 - how many boards?
24 - one round
25 - multiple rounds
26 - pause
27 - screen clear
28 - [ ] try a new board?
29   - [Life Lexicon Home Page](https://conwaylife.com/ref/
30     lexicon/lex_home.htm)
31
32 # `merge.c`
33
34 - examples
35 - for `merge-sort.c` later
36
37 # `insertion-sort.c`
38
39 - `for` + `while` version
40 - `for` + `for` version
```

```
1 add_executable(game-of-life game-of-life.c)
2 add_executable(game-of-life-chatgpt game-of-life-chatgpt.c
3 )
4 add_executable(insertion-sort insertion-sort.c)
5 add_executable(insertion-sort-bsearch insertion-sort-
6 bsearch.c)
7
8 add_executable(merge merge.c)
```

```
1 // Created by hfwei on 2024/10/16.
2
3 #include <stdio.h>
4 #include <unistd.h>
5 #include <stdlib.h>
6
7 #define SIZE 6
8
9 const int board[SIZE][SIZE] = {
10     { 0 },
11     { 0, 1, 1, 0, 0, 0 },
12     { 0, 1, 1, 0, 0, 0 },
13     { 0, 0, 0, 1, 1, 0 },
14     { 0, 0, 0, 1, 1, 0 },
15     { 0 }
16 };
17
18 //const int board[SIZE][SIZE] = {
19 //    [1][1] = 1, [1][2] = 1,
20 //    [2][1] = 1, [2][2] = 1,
21 //    [3][3] = 1, [3][4] = 1,
22 //    [4][3] = 1, [4][4] = 1
23 //};
24
25 int main(void) {
26     // TODO: play game-of-life
27     int old_board[SIZE + 2][SIZE + 2] = { 0 };
28
29     for (int row = 1; row <= SIZE; row++) {
30         for (int col = 1; col <= SIZE; col++) {
31             old_board[row][col] = board[row - 1][col - 1];
32         }
33     }
34
35     // print the initial board
36     for (int row = 1; row <= SIZE; row++) {
37         for (int col = 1; col <= SIZE; col++) {
38             printf("%c ", old_board[row][col] ? '*' : ' ');
39         }
40         printf("\n");
41     }
42     system("clear");
43
44     // old_board = apply the rule > new_board
```

```

45  int new_board[SIZE + 2][SIZE + 2] = { 0 };
46
47  for (int i = 0; i < 10; ++i) {
48      for (int row = 1; row <= SIZE; row++) {
49          for (int col = 1; col <= SIZE; col++) {
50              // counting live cells in the neighbour on
old_board[row][col]
51              int neighbors =
52                  old_board[row - 1][col - 1] +
53                  old_board[row - 1][col] +
54                  old_board[row - 1][col + 1] +
55                  old_board[row][col - 1] +
56                  old_board[row][col + 1] +
57                  old_board[row + 1][col - 1] +
58                  old_board[row + 1][col] +
59                  old_board[row + 1][col + 1];
60
61              // apply the rule
62              if (old_board[row][col]) {
63                  new_board[row][col] = (neighbors == 2 ||
neighbors == 3);
64              } else {
65                  new_board[row][col] = (neighbors == 3);
66              }
67          }
68      }
69
70      for (int row = 1; row <= SIZE; row++) {
71          for (int col = 1; col <= SIZE; col++) {
72              printf("%c ", new_board[row][col] ? '*' : ' ');
73          }
74          printf("\n");
75      }
76      // Linux: unistd.h
77      sleep(1);
78      // Windows: windows.h
79      // Sleep(1000);
80
81      // Linux: stdlib.h
82      system("clear");
83      // Window: stdlib.h
84      // system("cls");
85
86      // old_board <- new_board

```

```
87     for (int row = 1; row <= SIZE; row++) {
88         for (int col = 1; col <= SIZE; col++) {
89             old_board[row][col] = new_board[row][col];
90         }
91     }
92 }
93
94 return 0;
95 }
```

```
1 // Created by hfwei on 2024/10/16.
2 // Code generated by ChatGPT.
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <time.h>
7
8 #define MAX_LEN 10000
9 #define RANGE 100
10
11 int main(void) {
12     int numbers[MAX_LEN] = { 0 };
13
14     int size = 0;
15     scanf("%d", &size);
16
17     srand(time(NULL));
18     for (int i = 0; i < size; i++) {
19         numbers[i] = rand() % RANGE;
20     }
21
22     // print the original array
23     for (int i = 0; i < size; i++) {
24         printf("%d ", numbers[i]);
25     }
26     printf("\n");
27
28     // TODO: insertion sort
29     for (int i = 1; i < size; i++) {
30         // numbers[0 .. i - 1] are already sorted
31         // now consider key, the i-th element
32         int key = numbers[i];
33
34         // move elements > key
35         int j = i - 1;
36         while (j >= 0 && numbers[j] > key) {
37             numbers[j + 1] = numbers[j];
38             j = j - 1;
39         }
40
41         // a for-loop version
42         // for (j = i - 1; j >= 0 && numbers[j] > key; j--) {
43         //     numbers[j + 1] = numbers[j];
44         // }
```

```
45
46     // a wrong for-loop version
47 //     for (j = i - 1; j >= 0; j--) {
48 //         if (numbers[j] > key) {
49 //             numbers[j + 1] = numbers[j];
50 //         }
51 //     }
52
53     // put key there
54     numbers[j + 1] = key;
55
56     // now numbers[0 .. i] is already sorted
57     for (int j = 0; j < size; j++) {
58         printf("%d ", numbers[j]);
59     }
60     printf("\n");
61 }
62
63 // print the sorted array
64 for (int i = 0; i < size; i++) {
65     printf("%d ", numbers[i]);
66 }
67 printf("\n");
68
69 return 0;
70 }
```



```
1 // Created by hfwei on 2024/10/16.
2 // Code generated by ChatGPT.
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <time.h>
7 #include <unistd.h>
8
9 // Define grid dimensions
10 #define ROWS 20
11 #define COLS 40
12
13 // Function to initialize the grid randomly
14 void initializeGrid(int grid[ROWS][COLS]) {
15     for (int i = 0; i < ROWS; i++) {
16         for (int j = 0; j < COLS; j++) {
17             grid[i][j] = rand() % 2; // 0 (dead) or 1 (alive)
18         }
19     }
20 }
21
22 // Function to print the grid
23 void printGrid(int grid[ROWS][COLS]) {
24     for (int i = 0; i < ROWS; i++) {
25         for (int j = 0; j < COLS; j++) {
26             if (grid[i][j] == 1) {
27                 printf("#"); // Alive cell
28             } else {
29                 printf(" "); // Dead cell
30             }
31         }
32         printf("\n");
33     }
34     printf("\n");
35 }
36
37 // Function to update the grid for the next generation
38 void updateGrid(int grid[ROWS][COLS]) {
39     int newGrid[ROWS][COLS];
40
41     for (int i = 0; i < ROWS; i++) {
42         for (int j = 0; j < COLS; j++) {
43             int neighbors = 0;
```

```

45     // Count neighbors
46     for (int x = -1; x <= 1; x++) {
47         for (int y = -1; y <= 1; y++) {
48             if (x == 0 && y == 0) { continue; } // Skip the
current cell
49             int newX = i + x;
50             int newY = j + y;
51
52             if (newX >= 0 && newX < ROWS && newY >= 0 &&
newY < COLS) {
53                 neighbors += grid[newX][newY];
54             }
55         }
56     }
57
58     // Apply Game of Life rules
59     if (grid[i][j] == 1) {
60         newGrid[i][j] = (neighbors == 2 || neighbors == 3
) ? 1 : 0;
61     } else {
62         newGrid[i][j] = (neighbors == 3) ? 1 : 0;
63     }
64 }
65 }
66
67 // Update the grid
68 for (int i = 0; i < ROWS; i++) {
69     for (int j = 0; j < COLS; j++) {
70         grid[i][j] = newGrid[i][j];
71     }
72 }
73 }
74
75 int main(void) {
76     int grid[ROWS][COLS];
77
78     // Seed the random number generator with the current
time
79     srand(time(NULL));
80
81     // Initialize the grid
82     initializeGrid(grid);
83
84     // Number of generations

```

```
85  int generations = 50;
86
87  for (int gen = 0; gen < generations; gen++) {
88      system("clear"); // Use "clear" on Unix-based
        systems (Linux, macOS)
89      printf("Generation %d:\n", gen);
90      printGrid(grid);
91      updateGrid(grid);
92      sleep(1); // Sleep for 100ms
93  }
94
95  return 0;
96 }
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4
5 #define MAX_LEN 10000
6 #define RANGE 10
7
8 int main() {
9     int numbers[MAX_LEN] = { 0 };
10
11     int size = 0;
12     scanf("%d", &size);
13
14     srand(time(NULL));
15     for (int i = 0; i < size; i++) {
16         numbers[i] = rand() % RANGE;
17     }
18
19     // print the original array
20     for (int i = 0; i < size; i++) {
21         printf("%d ", numbers[i]);
22     }
23     printf("\n");
24
25     // TODO
26     for (int i = 1; i < size; i++) {
27         int key = numbers[i];
28         int low = 0;
29         int high = i - 1;
30
31         while (low <= high) {
32             int mid = (low + high) / 2;
33             if (key >= numbers[mid]) {
34                 low = mid + 1;
35             } else {
36                 high = mid - 1;
37             }
38         }
39
40         for (int j = i - 1; j >= low; j--) {
41             numbers[j + 1] = numbers[j];
42         }
43         numbers[low] = key;
44     }
```

```
45     for (int i = 0; i < size; i++) {
46         printf("%d ", numbers[i]);
47     }
48     printf("\n");
49 }
50
51 // Print the sorted array
52 for (int i = 0; i < size; i++) {
53     printf("%d ", numbers[i]);
54 }
55 printf("\n");
56
57 return 0;
58 }
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4
5 #define MAX_LEN 10000
6 #define RANGE 10
7
8 int main() {
9     int numbers[MAX_LEN] = { 0 };
10
11     int size = 0;
12     scanf("%d", &size);
13
14     srand(time(NULL));
15     for (int i = 0; i < size; i++) {
16         numbers[i] = rand() % RANGE;
17     }
18
19     // print the original array
20     for (int i = 0; i < size; i++) {
21         printf("%d ", numbers[i]);
22     }
23     printf("\n");
24
25     // insertion sort with binary search
26     for (int i = 1; i < size; i++) {
27         int key = numbers[i];
28         int low = 0;
29         int high = i - 1;
30
31         int pos = -1;
32         while (low <= high) {
33             int mid = (low + high) / 2;
34             if (key > numbers[mid]) {
35                 low = mid + 1;
36             } else if (key < numbers[mid]) {
37                 high = mid - 1;
38             } else {
39                 pos = mid;
40                 low = mid + 1;
41             }
42         }
43
44         if (pos == -1) {
```

```
45     pos = low;
46 } else {
47     pos++;
48 }
49
50 for (int j = i - 1; j >= pos; j--) {
51     numbers[j + 1] = numbers[j];
52 }
53
54 numbers[pos] = key;
55
56 for (int i = 0; i < size; i++) {
57     printf("%d ", numbers[i]);
58 }
59 printf("\n");
60 }
61
62 // print the sorted array
63 for (int i = 0; i < size; i++) {
64     printf("%d ", numbers[i]);
65 }
66 printf("\n");
67
68 return 0;
69 }
```