

```
1.  /**
2.   * fifteen.c
3.   *
4.   * CS50 AP
5.   * Name: Alexandra Pruenta
6.   *
7.   * Implements Game of Fifteen (generalized to d x d).
8.   *
9.   * Usage: fifteen d
10.  *
11.  * whereby the board's dimensions are to be d x d,
12.  * where d must be in [DIM_MIN,DIM_MAX]
13.  *
14.  * Note that usleep is obsolete, but it offers more granularity than
15.  * sleep and is simpler to use than nanosleep; `man usleep` for more.
16.  */
17.
18. // necessary for usleep
19. #define _XOPEN_SOURCE 500
20.
21. // libraries to include
22. #include <cs50.h>
23. #include <stdio.h>
24. #include <stdlib.h>
25. #include <unistd.h>
26.
27. // constants
28. #define DIM_MIN 3
29. #define DIM_MAX 9
30.
31. // globally declared board
32. int board[DIM_MAX][DIM_MAX];
33.
34. // globally declared board dimension
35. int dim;
36.
37. // prototypes
38. void clear(void);
39. void greet(void);
40. void init(void);
41. void draw(void);
42. bool move(int tile);
43. bool won(void);
44.
45. int main(int argc, string argv[])
46. {
47.     // 0. TODO Indicar el numero de argumentos que puede introducir
48.     if (argc != 2)
```

```
49.     {
50.         printf("Usage: fifteen dimension\n");
51.         return 1;
52.     }
53.
54.     // 1. TODO Dimensiones validas
55.     dim = atoi(argv[1]);
56.     if (dim < DIM_MIN || dim > DIM_MAX)
57.     {
58.         printf("Board must be between %i x %i and %i x %i, inclusive.\n",
59.             DIM_MIN, DIM_MIN, DIM_MAX, DIM_MAX);
60.         return 2;
61.     }
62.
63.     // open log file to record moves
64.     FILE* file = fopen("log.txt", "w");
65.     if (file == NULL)
66.     {
67.         return 3;
68.     }
69.
70.     // 2. TODO El usuario
71.     greet();
72.
73.     // 3. TODO Comenzar el cuadro
74.     init();
75.
76.     // accept moves until game is won
77.     while (true)
78.     {
79.         // 4. TODO Limpiar la pantalla
80.         clear();
81.
82.         // 5. TODO Dibujar como esta el cuadro
83.         draw();
84.
85.         // log the current state of the board (for testing)
86.         for (int i = 0; i < dim; i++)
87.         {
88.             for (int j = 0; j < dim; j++)
89.             {
90.                 fprintf(file, "%i", board[i][j]);
91.                 if (j < dim - 1)
92.                 {
93.                     fprintf(file, "|");
94.                 }
95.             }
96.             fprintf(file, "\n");
```

```
97.     }
98.     fflush(file);
99.
100.    // 6. TODO Revisar si gana
101.    if (won())
102.    {
103.        printf("ftw!\n");
104.        break;
105.    }
106.
107.    // 7. TODO Empezar a moverse
108.    printf("Tile to move: ");
109.    int tile = GetInt();
110.
111.    // quit if user inputs 0 (for testing)
112.    if (tile == 0)
113.    {
114.        break;
115.    }
116.
117.    // log move (for testing)
118.    fprintf(file, "%i\n", tile);
119.    fflush(file);
120.
121.    // 8. TODO Moverse si es posible, sino decir que es un "illegal move"
122.    if (!move(tile))
123.    {
124.        printf("\nIllegal move.\n");
125.        usleep(500000);
126.    }
127.
128.    // 9. TODO Reposar por el bien del juego
129.    usleep(500000);
130. }
131.
132. // close log
133. fclose(file);
134.
135. // 10. TODO Termina todo
136. return 0;
137. }
138.
139. /**
140.  * Clears screen using ANSI escape sequences.
141.  */
142. void clear(void)
143. {
144.     printf("\033[2J");
```

```
145.     printf("\033[%d;%dH", 0, 0);
146. }
147.
148. /**
149.  * Greets player.
150.  */
151. void greet(void)
152. {
153.     clear();
154.     printf("WELCOME TO GAME OF FIFTEEN\n");
155.     usleep(2000000);
156. }
157.
158. /**
159.  * Initializes the game's board with tiles numbered 1 through d*d - 1
160.  * (i.e., fills 2D array with values but does not actually print them).
161.  */
162. void init(void)
163. {
164.     // TODO
165.     // Los numeros del cuadro empezando en 0
166.     int lastnum = dim * dim - 1;
167.     for (int i = 0; i < dim; i++)
168.     {
169.         for (int j = 0; j < dim; j++)
170.         {
171.             board[i][j] = lastnum;
172.             lastnum--;
173.         }
174.     }
175.     // Cambiar siempre el dos y el uno
176.     if ((dim * dim - 1) % 2 != 0)
177.     {
178.         // El valor original de 2 que sea 1
179.         board[dim - 1][dim - 2] = 2;
180.         // El valor original de 1 que sea 2
181.         board[dim - 1][dim - 3] = 1;
182.     }
183. }
184.
185. /**
186.  * Prints the board in its current state.
187.  */
188. void draw(void)
189. {
190.     // TODO
191.     // Filas
192.     for(int i = 0; i < dim; i++)
```

```
193.     {
194.         // Columnas
195.         for (int j = 0; j < dim; j++)
196.         {
197.             if (board[i][j] == 0)
198.             {
199.                 printf("  _ ");
200.             }
201.
202.             else
203.             {
204.                 printf("%3d ", board[i][j]);
205.             }
206.         }
207.         printf("\n");
208.     }
209. }
210. /**
211.  * If tile borders empty space, moves tile and returns true, else
212.  * returns false.
213.  */
214. bool move(int tile)
215. {
216.     for (int i = 0; i < dim; i++)
217.     {
218.         // Columnas del cuadro
219.         for (int j = 0; j < dim; j++)
220.         {
221.             if (board[i][j] == tile)
222.             {
223.                 if ((i + 1 < dim) && (board[i + 1][j] == 0))
224.                 {
225.                     board[i + 1][j] = tile;
226.                     board[i][j] = 0;
227.                 }
228.                 else if ((i - 1 >= 0) && (board[i - 1][j] == 0))
229.                 {
230.                     board[i - 1][j] = tile;
231.                     board[i][j] = 0;
232.                 }
233.                 else if ((j + 1 < dim) && (board[i][j + 1] == 0))
234.                 {
235.                     board[i][j + 1] = tile;
236.                     board[i][j] = 0;
237.                 }
238.                 else if ((j - 1 >= 0) && (board[i][j - 1] == 0))
239.                 {
240.                     board[i][j - 1] = tile;
```

```
241.         board[i][j] = 0;
242.     }
243.     return true;
244. }
245. }
246. }
247.
248.     return false;
249. }
250.
251. /**
252.  * Returns true if game is won (i.e., board is in winning configuration),
253.  * else false.
254.  */
255. bool won(void)
256. {
257.     int n = 1;
258.     for (int i = 0; i < dim; i++)
259.     {
260.         for(int j = 0; j < dim; j++)
261.         {
262.             if (board[i][j] == n)
263.             {
264.                 n++;
265.                 if ((n == (dim * dim)) && (board[dim - 1][dim - 1] == 0))
266.                 {
267.                     return true;
268.                 }
269.             }
270.         }
271.     }
272.     return false;
273. }
```