

Universidad Panamericana  
Análisis y Diseño de Algoritmos



**Análisis y Diseño de Algoritmos  
Proyecto Final  
“El Laberinto del Minotauro”**

Prof.: Marco Antonio Galindo Corona

Integrantes:

- Santiago Sepúlveda Landeros
- Emilio Ignacio Romero Martínez
- Emilio Ramírez Martínez del Campo

Fecha de Entrega: 23 de Noviembre de 2021

# Módulos del Código

```
4  namespace PF_LaberintoMinotauro
5  {
6      class Program
7      {
8          static void Main(string[] args)
9          {
10             Random genSpawn = new Random();
11             int cont = 0;
12             int inic = 0;
13             char contb = 's';
14             int minox = genSpawn.Next(5, 39), chyx = genSpawn.Next(5, 39), chybx = genSpawn.Next(5, 39), chycx = genSpawn.Next(5, 39), ogrx = genSpawn.Next(5, 39),
15             int minoy = genSpawn.Next(3, 19), chyy = genSpawn.Next(3, 19), chby = genSpawn.Next(3, 19), chycy = genSpawn.Next(3, 19), ogry = genSpawn.Next(3, 19),
16             int[,] matrix = new int[40, 20];
17             int dif = 0;
18             do
19             {
20                 int punt = 0;
21                 string nombre=splash();
22                 int playerx = 1, playery = 1;
23                 matrix = new int[41, 21];
24                 modeSelect(ref matrix, ref playerx, ref playery, ref minox, ref minoy, ref chyx, ref chyy, ref chybx, ref chby, ref chycx, ref chcy, ref ogrx, ref
25                 } while (contb == 's' || contb == 'S');
26             }
27
28             static void readMaze(string mapa, ref int[,] matrix){...}
29             static void readPoints(){...}
30             static void printMaze(int[,] arr, int punt){...}
31             static string MapSelect(){...}
32             static void modeSelect(ref int[,] matrix, ref int playerx, ref int playery, ref int minox, ref int minoy, ref int chyx, ref int chyy, ref int chybx, ref
33             static void difficultySelect(ref int[,] matrix, ref int playerx, ref int playery, ref int minox, ref int minoy, ref int chyx, ref int chyy, ref int chybx, ref
34             static void control(ref int playerx, ref int playery, ref int[,] matrix, ref int cont, ref int punt, string nombre, int dif){...}
35             static void controlMin(ref int minox, ref int minoy, ref int[,] matrix){...}
36             static void controlChy(ref int chyx, ref int chyy, ref int[,] matrix){...}
37             static void controlOgr(ref int ogrx, ref int ogry, ref int[,] matrix){...}
38             static void controlPsy(ref int psyx, ref int psyy, ref int[,] matrix){...}
39
40             static void splayer(int playerx, int playery, ref int[,] matrix){...}
41             static void smino(int minox, int minoy, ref int[,] matrix){...}
42             static void schy(int chyx, int chyy, ref int[,] matrix){...}
43             static void sogr(int ogrx, int ogry, ref int[,] matrix){...}
44             static void sposy(int psyx, int psyy, ref int[,] matrix){...}
45
46             static void winScreen(ref int cont, int punt, string nombre, int dif){...}
47             static void gameOver(ref int cont){...}
48             static string splash(){...}
49             static void hold(){...}
50             static void saveGame(string saveFile, ref int[,] matrix, int punt){...}
51             static void writeScore(string nombre, int punt, string dif){...}
52             static void posResume(ref int[,] arr, ref int playerx, ref int playery, ref int minox, ref int minoy, ref int chyx, ref int chyy, ref int chybx, ref int chy
53             static void resumePoints(ref int punt){...}
54             static int askInt(string mes){...}
55             static void colisionCheck(int[,] matrix, int playerx, int playery, ref int cont){...}
56             static char s_or_n(string mes){...}
57             static string askStr(string mes){...}
58             static void spawnBlock(int[,] arr, ref int minox, ref int minoy, ref int chyx, ref int chyy, ref int chybx, ref int chby, ref int chycx, ref int chcy, ref
59
60         }
61     }
62 }
```

1. Declaración de Variables y Main
2. Selección y Configuración Previa al Comienzo de la Partida
3. Movimiento de los Personajes
4. Actualización de Posiciones de los Personajes en la Matriz
5. Operaciones Adicionales

## Explicación del Código por Módulos

### Módulo 1: “Declaración de Variables y Main”

```
8     static void Main(string[] args)
9     {
10        Random genSpawn = new Random();
11        int cont = 0;
12        int inic = 0;
13        char contb = 's';
14        int minox = genSpawn.Next(5, 39), chyx = genSpawn.Next(5, 39), chybx = genSpawn.Next(5, 39), chycx
15        int minoy = genSpawn.Next(3, 19), chyy = genSpawn.Next(3, 19), chyby = genSpawn.Next(3, 19), chcy
16        int[,] matrix = new int[40, 20];
17        int dif = 0;
18        do
19        {
20            int punt = 0;
21            string nombre=splash();
22            int playerx = 1, playery = 1;
23            matrix = new int[41, 21];
24            modeSelect(ref matrix, ref playerx, ref playery, ref minox, ref minoy, ref chyx, ref chyy, ref
25        } while (contb == 's' || contb == 'S');
26    }
```

- Este módulo es en donde se encuentra la función main de nuestro programa. Dentro de aquí es donde declaramos las variables que van a ser usadas a lo largo de todo nuestro juego.
- Creamos también un generador aleatorio de valores para poder generar las posiciones aleatorias en X y en Y de nuestros enemigos.
- La variable matrix es aquella que se usará para leer los diferentes mapas que hay dentro del juego.
- En realidad, todo el juego se corre dentro de “modeSelect”, tiene muchas funciones anidadas, por eso hay tantas variables que se usan.

### Módulo 2: “Selección y Configuración Previa al Comienzo de la Partida”

```
27
28     static void readMaze(string mapa, ref int[,] matrix)...
29
30     static void readPoints()...
31
32     static void printMaze(int[,] arr, int punt)...
33
34     static string MapSelect()...
35
36     static void modeSelect(ref int[,] matrix, ref int playerx, ref int playery, ref int
37
38     static void dificultySelect(ref int[,] matrix, ref int playerx, ref int playery, ref
```

- Dentro de este módulo, se realizan todas las preparaciones y operaciones necesarias para realizar el juego.
- El método “readMaze” se encarga de convertir los strings que se leen de los archivos en valores numéricos de tipo int, con ayuda del separador de las comas. De esta manera los valores se pueden meter en el arreglo de matrix ya de forma numérica.

```

28     static void readMaze(string mapa, ref int[,] matrix)
29     {
30         string[] lineas = new string[20];
31         char[] div = { ' ', ',' };
32         StreamReader objleer = new StreamReader(mapa);
33         string linea = objleer.ReadLine();
34         while (linea != null)
35         {
36             for (int current = 0; current < matrix.GetLength(1)-1; current++)
37             {
38
39                 lineas = linea.Split(div);
40                 for (int i = 0; i < matrix.GetLength(0)-1; i++)
41                 {
42                     matrix[i, current] = Convert.ToInt32(lineas[i]);
43                 }
44                 linea = objleer.ReadLine();
45             }
46         }
47     }
48 }
```

- “readPoints” muestra los puntajes anteriores de todos los jugadores que han jugado el juego (es una de las 3 opciones del menú principal). Esto lo hace leyendo un archivo en donde se encuentran guardados el nombre del jugador, el puntaje que alcanzó, y la dificultad en que lo hizo.

```

49     static void readPoints()
50     {
51         StreamReader objleer = new StreamReader("Puntaje.txt");
52         string item = objleer.ReadLine();
53         Console.WriteLine("Nombre\t Puntaje\tDificultad\n");
54         while (item != null)
55         {
56             Console.WriteLine(item);
57             item = objleer.ReadLine();
58         }
59     }
60 }
```

- “printMaze” convierte los archivos en donde se encuentran guardados laberintos en forma de números, a diferentes formas y figuras.
- Para el laberinto, los 1 representan una pared, y los 0 representan un espacio libre. El 4 representa la salida y el 3 los dracmas que se pueden recolectar para un puntaje extra. Del 5 al 9 representan los diferentes personajes.

```

61     static void printMaze(int[,] arr, int punt)
62     {
63         Console.Clear();
64         for (int j = 0; j < arr.GetLength(1); j++)
65         {
66             for (int i = 0; i < arr.GetLength(0); i++)
67             {
68                 if (arr[i, j] == 1)
69                 {
70                     Console.ForegroundColor = ConsoleColor.Black;
71                     Console.BackgroundColor = ConsoleColor.DarkYellow;
72                     Console.Write("#");
73                     Console.BackgroundColor = ConsoleColor.Black;
74                     Console.ForegroundColor = ConsoleColor.White;
75                 }
76                 if (arr[i, j] == 0)
77                 {
78                     Console.Write(" ");
79                 }
80                 if (arr[i, j] == 2)
81                 {
82                     Console.Write("=");
83                 }
84                 if (arr[i, j] == 3)
85                 {
86                     Console.ForegroundColor = ConsoleColor.Yellow;
87                     Console.Write("+");
88                     Console.ForegroundColor = ConsoleColor.White;
89                 }
90                 if (arr[i, j] == 4)

```

- “MapSelect” simplemente se encarga de mostrar las diferentes opciones de mapa que están disponibles para el juego, y de pedirle al jugador una elección de mapa.

```

161     static string MapSelect()
162     {
163         Console.Clear();
164         Console.ForegroundColor = ConsoleColor.DarkCyan;
165         StreamReader objleer = new StreamReader("SMTitulo.txt");
166         string minotauro = objleer.ReadLine();
167         while (minotauro != null)
168         {
169             for (int current = 0; current <= 29; current++)
170             {
171                 Console.WriteLine(minotauro);
172                 minotauro = objleer.ReadLine();
173             }
174         }
175         objleer.Close();
176         Console.ForegroundColor = ConsoleColor.White;
177
178         int mapselec = askint("Ingrese su selección");
179
180         switch (mapselec)
181         {
182             case 1:
183                 string opcion = "mapauno.txt";
184                 return opcion;
185
186             case 2:
187                 opcion = "mapados.txt";
188                 return opcion;
189
190             case 3:
191                 opcion = "mapatres.txt";
192                 return opcion;
193
194             case 4:
195                 opcion = "mapacuatro.txt";
196                 return opcion;
197
198             case 5:
199                 opcion = "mapacinco.txt";
200                 return opcion;
201             default:
202                 opcion = "mapauno.txt";
203                 return opcion;
204
205
206         }
207     }

```

- “modeSelect”, como se había dicho anteriormente, es la método en donde corre todo el programa.
- Esto es debido a que es la primera elección que se le presenta al jugador, y debido a lo que se escoja, son los diferentes procedimiento que se van a ejecutar (jugar una nueva partida, cargar una partida guardada, o ver las puntuaciones).

```

208     static void modoSelect(ref int[,] matrix, ref int playerx, ref int playery, ref int minox, ref int minoy, ref int chyx, ref int chby, ref int chbx)
209    {
210        Console.Clear();
211        ConsoleColor foregroundColor = ConsoleColor.DarkRed;
212        StreamReader objleer = new StreamReader("Smodo.txt");
213        string minotauro = objleer.ReadLine();
214        while (minotauro != null)
215        {
216            for (int current = 0; current <= 22; current++)
217            {
218
219                Console.WriteLine(minotauro);
220                minotauro = objleer.ReadLine();
221            }
222        }
223        objleer.Close();
224        ConsoleColor foregroundColor = ConsoleColor.White;
225        inic = askInt("Seleccione una opción");
226        switch (inic)
227        {
228            case 1:
229                string mapa = MapSelect();
230                readMaze(mapa, ref matrix);
231                spawnBlock(matrix, ref minox, ref minoy, ref chyx, ref chyy, ref chbx, ref chby, ref chcx, ref chcy, ref ogrx, ref
232                dificultySelect(ref matrix, ref playerx, ref playery, ref minox, ref minoy, ref chyx, ref chyy, ref chbx, ref chby,
233                contb = s_or_n("¿Quieres jugar de nuevo? ");
234                Console.Clear();
235                break;
236            case 2:
237                readMaze("SavedGame.txt", ref matrix);
238                resumePoints(ref punt);
239                posResume(ref matrix, ref playerx, ref playery, ref minox, ref minoy, ref chyx, ref chyy, ref chbx, ref chby, ref ch
240                dificultySelect(ref matrix, ref playerx, ref playery, ref minox, ref minoy, ref chyx, ref chyy, ref chbx, ref chby,
241                contb = s_or_n("¿Quieres jugar de nuevo? ");
242                Console.Clear();
243                break;
244            case 3:
245                readPoints();
246                contb = s_or_n("¿Quieres volver al menú principal? ");
247                break;
248        }
249    }

```

- “difficultySelect” le presenta las opciones de dificultad al jugador por medio de un archivo, y posteriormente le pide que seleccione una dificultad. La dificultad varía por medio del número de enemigos.
- Si se elige la primera opción, sólamente se convocan los métodos de 3 enemigos, uno de cada tipo, además de la función del personaje del jugador.
- Si se elige la segunda opción, se convocan los métodos de 6 enemigos, dos de cada tipo, además de la función del personaje del jugador.
- Si se elige la tercera opción, se convocan los métodos de 9 enemigos, tres de cada tipo, además de la función del personaje del jugador.

```

295     case 1:
296         while (cont == 0)
297     {
298             printMaze(matrix, punt);
299
300             control(ref playerx, ref playery, ref matrix, ref cont, ref punt, nombre, dif);
301             controlMin(ref minox, ref minoy, ref matrix);
302             controlChy(ref chyx, ref chyy, ref matrix);
303             controlOgr(ref ogrx, ref ogry, ref matrix);
304             controlPsy(ref psyx, ref psyy, ref matrix);
305             colisionCheck(matrix, playerx, playery, ref cont);
306         }
307
308         break;
309
310     case 2:
311         while (cont == 0)
312     {
313             printMaze(matrix, punt);
314
315             control(ref playerx, ref playery, ref matrix, ref cont, ref punt, nombre, dif);
316             controlMin(ref minox, ref minoy, ref matrix);
317             controlChy(ref chyx, ref chyy, ref matrix);
318             controlChy(ref chybx, ref chyby, ref matrix);
319             controlOgr(ref ogrx, ref ogry, ref matrix);
320             controlOgr(ref ogrbx, ref ogrby, ref matrix);
321             controlPsy(ref psyx, ref psyy, ref matrix);
322             controlPsy(ref psybx, ref psyby, ref matrix);
323             colisionCheck(matrix, playerx, playery, ref cont);
324         }
325
326         break;
327
328     case 3:
329         while (cont == 0)
330     {
331             printMaze(matrix, punt);
332
333             control(ref playerx, ref playery, ref matrix, ref cont, ref punt, nombre, dif);
334         }
335
336         break;
337
338     case 4:
339         while (cont == 0)
340     {
341             printMaze(matrix, punt);
342
343             control(ref playerx, ref playery, ref matrix, ref cont, ref punt, nombre, dif);
344         }
345

```

### Módulo 3: “Movimiento de los Personajes”

```

368     static void control(ref int playerx, ref int playery, ref int[,] matrix, ref int cont, ref int punt, string nombre, int dif)...
369     static void controlMin(ref int minox, ref int minoy, ref int[,] matrix)...  

400     static void controlChy(ref int chyx, ref int chyy, ref int[,] matrix)...  

507     static void controlOgr(ref int ogrx, ref int ogry, ref int[,] matrix)...  

554     static void controlPsy(ref int psyx, ref int psyy, ref int[,] matrix)...  

601
648

```

- Este módulo controla la manera en que se mueven los personajes dentro del laberinto.
- Existe una función que controla los movimientos del personaje principal, en función a la tecla que se oprima, y otras funciones que controlan el movimiento aleatorio de los enemigos.

- El método control revisa los valores posibles que se tienen en la matriz según la tecla que se oprime, por medio de un switch.
- Por ejemplo, si se oprime la tecla “RightArrow”, se revisan los valores que se tienen a la derecha del jugador.
- Si el valor es 0 (un espacio dentro del laberinto), entonces se incrementa el valor de la posición en X del personaje principal.
- Si el valor es 3 (una moneda dentro del laberinto), se incrementa el valor de la posición en X del personaje principal y además se añaden 10 puntos al contador del puntaje.
- Si el valor es 4 (la salida dentro del laberinto), entonces se corre el método de cuando se ha ganado la partida.
- Lo mismo sucede con las teclas de izquierda, arriba y abajo, con las posiciones respectivas en X y Y.
- En el caso de que se oprime la tecla S, se guarda la matriz de la partida con los valores actuales en un archivo por medio del método “saveGame”.

```

369
370     static void control(ref int playerx, ref int playery, ref int[,] matrix, ref int cont, ref int punt, string nombre, int dif)
371     {
372         ConsoleKeyInfo input;
373         input = Console.ReadKey(false);
374
375         switch (input.Key)
376         {
377             case ConsoleKey.RightArrow:
378                 if (matrix[playerx + 1, playery] == 0 || matrix[playerx + 1, playery] == 3)
379                 {
380                     matrix[playerx, playery] = 0;
381                     playerx++;
382                     punt++;
383                     if (matrix[playerx, playery] == 3)
384                     {
385                         punt += 10;
386                     }
387                     splayer(playerx, playery, ref matrix);
388                 }
389                 if (matrix[playerx + 1, playery] == 4)
390                 {
391                     winScreen(ref cont, punt, nombre, dif);
392                 }
393                 break;
394
395             case ConsoleKey.LeftArrow:
396                 if (matrix[playerx - 1, playery] == 0 || matrix[playerx - 1, playery] == 3)
397                 {
398                     matrix[playerx - 1, playery] = 0;
399
400

```

- Las funciones “controlMin”, “controlChy”, “controlOgr” y “controlPsy”, hacen esencialmente lo mismo que la función de “control”.
- Genera un número aleatorio que va del 1 al 4, y dentro de un switch, se dan los movimientos aleatorios dependiendo del número aleatorio que se haya generado.
- Revisa que haya un espacio a la derecha (es decir, un valor de 0 en el arreglo de matrix), y después aumenta o disminuye por una unidad el valor de la posición de los enemigos en dirección horizontal o vertical, dependiendo del caso. Después corre los métodos que actualizan las posiciones dentro de la matriz.

```

463     int randirmin = genAleatorio.Next(1, 5);
464
465     switch (randirmin)
466     {
467         case 1:
468             if (matrix[minox + 1, minoy] == 0)
469             {
470                 matrix[minox, minoy] = 0;
471                 minox++;
472                 smino(minox, minoy, ref matrix);
473             }
474             break;
475         case 2:
476             if (matrix[minox - 1, minoy] == 0)
477             {
478                 matrix[minox, minoy] = 0;
479             }
480     }
481 }
482
507     static void controlChy(ref int chyx, ref int chyy, ref int[,] matrix)
508     {
509         Random genAleatorio = new Random();
510         int randirmin = genAleatorio.Next(1, 5);
511
512         switch (randirmin)
513         {
514             case 1:
515                 if (matrix[chyx + 1, chyy] == 0)
516                 {
517                     matrix[chyx, chyy] = 0;
518                     chyx++;
519                     schy(chyx, chyy, ref matrix);
520                 }
521                 break;
522             case 2:
523                 if (matrix[chyx - 1, chyy] == 0)
524                 {
525                     matrix[chyx, chyy] = 0;
526                 }
527         }
528     }
529
544     static void controlOgr(ref int ogrx, ref int ogry, ref int[,] matrix)
545     {
546         Random genAleatorio = new Random();
547         int randirmin = genAleatorio.Next(1, 5);
548
549         switch (randirmin)
550         {
551             case 1:
552                 if (matrix[ogrx + 1, ogry] == 0)
553                 {
554                     matrix[ogrx, ogry] = 0;
555                     ogrx++;
556                     sogr(ogrx, ogry, ref matrix);
557                 }
558                 break;
559             case 2:
560                 if (matrix[ogrx - 1, ogry] == 0)
561
562         static void controlPsy(ref int psyx, ref int psyy, ref int[,] matrix)
563         {
564             Random genAleatorio = new Random();
565             int randirmin = genAleatorio.Next(1, 5);
566
567             switch (randirmin)
568             {
569                 case 1:
570                     if (matrix[psyx + 1, psyy] == 0)
571                     {
572                         matrix[psyx, psyy] = 0;
573                         psyx++;
574                         spsy(psyx, psyy, ref matrix);
575                     }
576                     break;
577                 case 2:
578                     if (matrix[psyx - 1, psyy] == 0)
579                     {
580                         matrix[psyx, psyy] = 0;
581                     }
582             }
583         }

```

## Módulo 4: “Actualización de Posiciones de los Personajes en la Matriz”

```
648     static void $player(int playerx, int playery, ref int[,] matrix)
649     {
650         matrix[playerx, playery] = 5;
651     }
652     static void $mino(int minox, int minoy, ref int[,] matrix)
653     {
654         matrix[minox, minoy] = 6;
655     }
656     static void $chy(int chyx, int chyy, ref int[,] matrix)
657     {
658         matrix[chyx, chyy] = 7;
659     }
660     static void $ogr(int ogrx, int ogry, ref int[,] matrix)
661     {
662         matrix[ogrx, ogry] = 8;
663     }
664     static void $psy(int psyx, int psyy, ref int[,] matrix)
665     {
666         matrix[psyx, psyy] = 9;
667     }
668 }
669 }
```

- Este módulo en realidad es bastante simple.
- Las funciones que se tienen aquí son las que se invocan en el módulo anterior para actualizar el movimiento de las posiciones de los personajes dentro de la matriz.
- En los valores de posición, que se movieron en el módulo anterior, se escribe al número que representa a cada personaje, y así se actualizan los valores de la matriz de forma numérica.

## Módulo 5: “Operaciones Adicionales”

```
669     static void winScreen(ref int cont, int punt, string nombre, int dif)...
670     static void gameOver(ref int cont)...
671     static string splash()...
672     static void hld()...
673     static void saveGame(string saveFile, ref int[,] matrix, int punt)...
674     static void writeScore(string nombre, int punt, string dif)...
675     static void ppsResume(ref int[,] arr, ref int playerx, ref int playery, ref int minox, ref int minoy, ref int chyx, ref int chyy)
676     static void resumePoints(ref int punt)...
677     static int gskint(string mes)...
678     static void colisionCheck(int[,] matrix, int playerx, int playery, ref int cont)...
679     static char sOr_n(string mes)...
680     static string askstr(string mes)...
681     static void spawnBlock(int[,] arr, ref int minox, ref int minoy, ref int chyx, ref int chyy, ref int chyb, ref int chby, ref int chbx, ref int chbyx)
```

- “winScreen” es la pantalla que aparece una vez que se consigue pasar el nivel, esencialmente lo mismo que la función de “control”.
  - Nos aparece el nombre que el usuario ingresó, el nivel de dificultad que fue seleccionado, al igual que la cantidad de puntos que se consiguieron durante el juego. Todos estos datos son guardados en un archivo

- “gameOver” es la pantalla que aparece si una de las criaturas nos atrapa.
  - Nos muestra una opción para que podamos decidir si queremos volver a jugar o no.

- “Splash” es la función que nos pide el nombre para empezar a jugar.
  - Este lo guarda en los archivos para después devolverlo con las puntuación y su dificultad

```
744 static string splash() {
745     StreamWriter objescrib = new StreamWriter("Dif.txt", true);
746     objescrib.Close();
747     Console.ForegroundColor = ConsoleColor.DarkYellow;
748     StreamReader objleer = new StreamReader("MinoArt.txt");
749     string minotauro = objleer.ReadLine();
750     while (minotauro != null)
751     {
752         for (int current = 0; current <= 49; current++)
753         {
754             Console.WriteLine(minotauro);
755             minotauro = objleer.ReadLine();
756         }
757     }
758     objleer.Close();
759     Console.ForegroundColor = ConsoleColor.White;
760     string nombre = askstr("Nombre del Jugador: ");
761     return nombre;
762
763
764
```

- “saveGame” se guarda el progreso que se lleva del juego en un archivo
- Después en el menú principal existe una opción “continuar partida”, si la seleccionamos nos va a llevar al juego que nosotros guardamos con esta función.

```

773 1 referencia
773     static void saveGame(string saveFile, ref int[,] matrix, int punt)
774     {
775         File.WriteAllText(@"SavedGame.txt", string.Empty);
776         StreamWriter objgrabar = new StreamWriter(saveFile, true);
777
778         for (int i = 0; i < matrix.GetLength(1)-1; i++)
779         {
780             for (int j = 0; j < matrix.GetLength(0)-1; j++)
781             {
782                 string value = Convert.ToString(matrix[j, i]);
783                 objgrabar.Write("{0}, ", value);
784             }
785             objgrabar.WriteLine();
786         }
787         objgrabar.Close();
788
789         File.WriteAllText(@"ScoreSave.txt", string.Empty);
790         StreamWriter objgrabarb = new StreamWriter("ScoreSave.txt", true);
791         objgrabarb.Write(punt);
792         objgrabarb.Close();
793
794
795
    
```

- “writeScore” Se escribe el nombre ingresado al inicio, el puntaje obtenido durante la partida, así como el nivel de dificultad que fue seleccionado

```

795 1 referencia
795     static void writeScore(string nombre, int punt, string dif)
796     {
797         StreamWriter objgrabar = new StreamWriter("Puntaje.txt", true);
798         objgrabar.WriteLine("{0}\t{1}\t{2}", nombre, punt, dif);
799         objgrabar.WriteLine("\n");
800         objgrabar.Close();
801
802     }
    
```

- “posResume” usa los valores de posición de los personajes que fueron guardados en la partida
- Para poder mostrarlos una vez que se reanude el juego

```

803     a referencia
804     static void posResume(ref int[,] arr, ref int playerx, ref int playery, ref int minox, ref int minoy, ref int chyx, ref int
805     {
806         Console.Clear();
807         for (int j = 0; j < arr.GetLength(1); j++)
808         {
809             for (int i = 0; i < arr.GetLength(0); i++)
810             {
811                 if (arr[i, j] == 5)
812                 {
813                     playerx = i;
814                     playery = j;
815                     arr[i, j] = 0;
816                 }
817                 if (arr[i, j] == 6)
818                 {
819                     minox = i;
820                     minoy = j;
821                     arr[i, j] = 0;
822                 }
823                 if (arr[i, j] == 7)
824                 {
825                     chyx = i;
826                     chyy = j;
827                     arr[i, j] = 0;
828                 }
829                 if (arr[i, j] == 7)
830                 {
831                     chybx = i;
832                     chby = j;
833                     arr[i, j] = 0;
834                 }
835                 if (arr[i, j] == 7)
836                 {
837                     chycx = i;
838                     chcy = j;
839                     arr[i, j] = 0;
840                 }
841             }
842         }
843     }
844     if (arr[i, j] == 8)
845     {
846         ogrx = i;
847         ogry = j;
848         arr[i, j] = 0;
849     }
850     if (arr[i, j] == 8)
851     {
852         ogrbx = i;
853         ogrby = j;
854         arr[i, j] = 0;
855     }
856     if (arr[i, j] == 8)
857     {
858         ogrcx = i;
859         ogrcy = j;
860         arr[i, j] = 0;
861     }
862     if (arr[i, j] == 9)
863     {
864         psyx = i;
865         psyy = j;
866         arr[i, j] = 0;
867     }
868     if (arr[i, j] == 9)
869     {
870         psybx = i;
871         psyby = j;
872         arr[i, j] = 0;
873     }
874     if (arr[i, j] == 9)
875     {
876         psycx = i;
877         psocy = j;
878         arr[i, j] = 0;
879     }
880 
```

- “resumePoints” lee los datos de la puntuación de la partida que fueron guardados en el archivo

```

892     1 referencia
892     static void resumePoints(ref int punt)
893     {
894         StreamReader objleerb = new StreamReader("ScoreSave.txt");
895         punt = Convert.ToInt32(objleerb.ReadLine());
896         objleerb.Close();
897     }
897     3 referencias

```

- “Askint” pide un mensaje de enteros
- Pide un entero al usuario

```

898     3 referencias
898     static int askint(string mes)
899     {
900         int output;
901         Console.WriteLine("{0}", mes);
902         output = Convert.ToInt32(Console.ReadLine());
903         return output;
904     }
904     3 referencias

```

- “colisionCheck” si el jugador se encuentra en la misma posición o a un espacio de distancia que cualquiera de las 4 criaturas el juego estará acabado.
- Por consiguiente aparecerá la pantalla de gameOver

```

905     3 referencias
905     static void colisionCheck(int[,] matrix, int playerx, int playery, ref int cont)
906     {
907         if (matrix[playerx + 1, playery] == 6 || matrix[playerx + 1, playery] == 7 || matrix[playerx + 1, playery] == 8 || matrix[playerx + 1, playery] == 9)
908         {
909             gameOver(ref cont);
910         }
911         if (matrix[playerx - 1, playery] == 6 || matrix[playerx - 1, playery] == 7 || matrix[playerx - 1, playery] == 8 || matrix[playerx - 1, playery] == 9)
912         {
913             gameOver(ref cont);
914         }
915         if (matrix[playerx, playery + 1] == 6 || matrix[playerx, playery + 1] == 7 || matrix[playerx, playery + 1] == 8 || matrix[playerx, playery + 1] == 9)
916         {
917             gameOver(ref cont);
918         }
919         if (matrix[playerx, playery - 1] == 6 || matrix[playerx, playery - 1] == 7 || matrix[playerx, playery - 1] == 8 || matrix[playerx, playery - 1] == 9)
920         {
921             gameOver(ref cont);
922         }
923     }
923     3 referencias

```

- “S\_or\_n” es el mensaje que aparece para saber si se desea continuar o no.

```
924     static char s_or_n(string mes)
925     {
926         Console.WriteLine("");
927         Console.WriteLine("{0} (s/n) <enter>", mes);
928         char output = Convert.ToChar(Console.ReadLine());
929         Console.Clear();
930         return output;
931     }
932 }
```

1 referencia

- “Asktr” pide un mensaje de enteros para que sea escrito
- Pide un string al usuario

```
933     static string askstr(string mes)
934     {
935         string output;
936         Console.WriteLine("{0}", mes);
937         output = Console.ReadLine();
938         return output;
939 }
```

- “spawnBlock” es el rango en “x” y “y” donde pueden aparecer todas las criaturas.

```

940
941
942
943     Random genSpawn = new Random();
944
945     if (arr[minox, minoy] != 0)
946     {
947         minox = genSpawn.Next(1, 40);
948         minoy = genSpawn.Next(1, 20);
949     }
950
951
952     if (arr[chyx, chyy] != 0)
953     {
954         chyx = genSpawn.Next(1, 40);
955         chyy = genSpawn.Next(1, 20);
956     }
957
958
959     if (arr[chybx, chby] != 0)
960     {
961         chybx = genSpawn.Next(1, 40);
962         chby = genSpawn.Next(1, 20);
963     }
964
965
966     if (arr[chycx, chcy] != 0)
967     {
968         chycx = genSpawn.Next(1, 40);
969         chcy = genSpawn.Next(1, 20);
970     }
971
972
973     if (arr[ogrx, ogry] != 0)
974     {
975         ogrx = genSpawn.Next(1, 40);
976         ogry = genSpawn.Next(1, 20);
977     }
978
979
980     if (arr[ogrby, ogrbx] != 0)
981     {
982         ogrbx = genSpawn.Next(1, 40);
983         ogrby = genSpawn.Next(1, 20);
984     }
985
986
987     if (arr[ogrcx, ogrcy] != 0)
988     {
989         ogrcx = genSpawn.Next(1, 40);
990         ogrcy = genSpawn.Next(1, 20);
991     }
992
993
994     if (arr[psyx, psyy] != 0)
995     {
996         psyx = genSpawn.Next(2, 39);
997         psyy = genSpawn.Next(2, 19);
998     }
999
1000
1001     if (arr[psybx, psyby] != 0)
1002     {
1003         psybx = genSpawn.Next(2, 39);
1004         psyby = genSpawn.Next(2, 19);
1005     }
1006
1007
1008     if (arr[psycx, psycy] != 0)
1009     {
1010         psycx = genSpawn.Next(2, 39);
1011         psycy = genSpawn.Next(2, 19);
1012     }
1013
1014

```

## Instrucciones del Juego

Instrucciones antes de comenzar:

- Primero debes escribir tu nombre, para poder guardar tu puntuación en caso de que ganes la partida.
- Posteriormente, debes de elegir entre empezar una nueva partida, continuar con una partida guardada (en caso de que se tenga), o ver las puntuaciones que se han hecho en el juego.
- Ya que se ha seleccionado la opción de comenzar una nueva partida, debes de elegir uno de los 5 mapas disponibles para jugar tu partida.
- Despues, debes de elegir la dificultad con la que quieras jugar:
  - Fácil: 3 enemigos
  - Medio: 6 enemigos
  - Difícil: 9 enemigos

Reglas del juego:

1. Para iniciar el juego, debes de oprimir la flecha derecha de tu ordenador.
2. Los movimientos de tu personaje son controlados por medio de las flechas de tu ordenador.
3. Sólamente puedes moverte un espacio a la vez.
4. Debes de cruzar todo el laberinto, y tratar de llegar a la salida, que está de color verde.
5. Hay enemigos distribuidos aleatoriamente en todo el laberinto, cuyo movimiento también es aleatorio.
6. El movimiento de los enemigos también se limita a un espacio por turno.
7. Si te llegas a encontrar directamente a lado de uno de los enemigos, es decir, un espacio a su izquierda, derecha, arriba o abajo, habrás perdido.
8. Cada espacio que tu personaje se mueva sin llegar a estar a lado de un enemigo, aumentará un punto en tu puntaje.
9. Hay dracmas distribuidos a lo largo del laberinto (+), los cuales puedes agarrar para sumar 10 puntos adicionales a tu puntaje.
10. La partida termina cuando cruzas la salida del laberinto.

# Código Completo

```
//Declaración de variables y la manera aleatoria en que se mueven las criaturas del laberinto
static void Main(string[] args)
{
    Random genSpawn = new Random();
    int cont = 0;
    int inic = 0;
    char contb = 's';
    int minox = genSpawn.Next(5, 39), chyx = genSpawn.Next(5, 39), chybx = genSpawn.Next(5, 39), chycx = genSpawn.Next(5, 39), ogrx = genSpawn.Next(5, 39), ogrbx
= genSpawn.Next(5, 39), ogrcx = genSpawn.Next(5, 39), psyx = genSpawn.Next(5, 39), psybx = genSpawn.Next(5, 39), psycx = genSpawn.Next(5, 39),
    int minoy = genSpawn.Next(3, 19), chyy = genSpawn.Next(3, 19), chyby = genSpawn.Next(3, 19), chycy = genSpawn.Next(3, 19), ogry = genSpawn.Next(3, 19), ogrby
= genSpawn.Next(3, 19), ogrcy = genSpawn.Next(3, 19), psyy = genSpawn.Next(3, 19), psyby = genSpawn.Next(3, 19), psycy = genSpawn.Next(3, 19);
    int[,] matrix = new int[40, 20];
    int dif = 0;
    do
    {
        int punt = 0;
        string nombre=splash();
        int playerx = 1, playery = 1;
        matrix = new int[41, 21];
        modeSelect(ref matrix, ref playerx, ref playery, ref minox, ref minoy, ref chyx, ref chyy, ref chybx, ref chyby, ref chycx, ref chycy, ref ogrx, ref ogry, ref ogrbx,
ref ogrcx, ref ogrcy, ref psyx, ref psybx, ref psyby, ref psycx, ref psycy, ref cont, ref punt, ref contb, ref inic, ref dif, nombre);
        } while (contb == 's' || contb == 'S');
    }

//Lee los valores del mapa de un archivo y los mete en un arreglo bidimensional
static void readMaze(string mapa, ref int[,] matrix)
{
    string[] lineas = new string[20];
    char[] div = {'\r'};
    StreamReader objleer = new StreamReader(mapa);
    string linea = objleer.ReadLine();
    while (linea != null)
    {
        for (int current = 0; current < matrix.GetLength(1)-1; current++)
        {

            lineas = linea.Split(div);
            for (int i = 0; i < matrix.GetLength(0)-1; i++)
            {
                matrix[i, current] = Convert.ToInt32(lineas[i]);
            }
            linea = objleer.ReadLine();
        }
        objleer.Close();
    }
}

//Saca los valores de puntaje de los jugadores del archivo donde están guardados y los despliega al usuario.
static void readPoints()
{
    StreamReader objleer = new StreamReader("Puntaje.txt");
    string item = objleer.ReadLine();
    Console.WriteLine("Nombre\t Puntaje\tDificultad\n");
    while (item != null)
    {
        Console.WriteLine(item);
        item = objleer.ReadLine();
    }
    objleer.Close();
}

//Usando el arreglo bidimensional de datos y dependiendo de los números, se despliega un elemento gráfico distinto en la terminal.
static void printMaze(int[,] arr, int punt)
{
    Console.Clear();
    for (int j = 0; j < arr.GetLength(1); j++)
    {
        for (int i = 0; i < arr.GetLength(0); i++)
        {
            if (arr[i, j] == 1)
            {
                Console.ForegroundColor = ConsoleColor.Black;
                Console.BackgroundColor = ConsoleColor.DarkYellow;
                Console.Write("█");
                Console.BackgroundColor = ConsoleColor.Black;
                Console.ForegroundColor = ConsoleColor.White;
            }
        }
    }
}
```

```

        }
        if (arr[i, j] == 0)
        {
            Console.Write(" ");
        }
        if (arr[i, j] == 2)
        {
            Console.Write("—");
        }
        if (arr[i, j] == 3)
        {
            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.Write("+");
            Console.ForegroundColor = ConsoleColor.White;
        }
        if (arr[i, j] == 4)
        {
            Console.BackgroundColor = ConsoleColor.DarkGreen;
            Console.Write(" ");
            Console.BackgroundColor = ConsoleColor.Black;
        }
        if (arr[i, j] == 5)
        {
            Console.ForegroundColor = ConsoleColor.White;
            Console.Write("Σ");
            Console.ForegroundColor = ConsoleColor.White;
        }
        if (arr[i, j] == 6)
        {
            Console.ForegroundColor = ConsoleColor.Red;
            Console.Write("π");
            Console.ForegroundColor = ConsoleColor.White;
        }
        if (arr[i, j] == 7)
        {
            Console.ForegroundColor = ConsoleColor.Magenta;
            Console.Write("$");
            Console.ForegroundColor = ConsoleColor.White;
        }
        if (arr[i, j] == 8)
        {
            Console.ForegroundColor = ConsoleColor.Blue;
            Console.Write("φ");
            Console.ForegroundColor = ConsoleColor.White;
        }
        if (arr[i, j] == 9)
        {
            Console.ForegroundColor = ConsoleColor.DarkGray;
            Console.Write("Θ");
            Console.ForegroundColor = ConsoleColor.White;
        }
    }
    Console.WriteLine("\n");
}

ConsoleCursorPosition(0, 21);
Console.WriteLine("Puntuación: {0}\n\n", punt);
Console.WriteLine("Instrucciones: ");
Console.WriteLine("1. Presione '→' para comenzar la partida");
Console.WriteLine("2. Presione 'S' para guardar la partida");
Console.WriteLine("3. Intenta llegar a la salida sin tocar a los enemigos");
Console.WriteLine("4. Agarrar Dracmas (+) te da 10 puntos extras");
Console.WriteLine("5. Si te encuentras directamente junto a un enemigo, habrás perdido la partida");
Console.SetCursorPosition(45, 5);
Console.ForegroundColor = ConsoleColor.White;
Console.SetCursorPosition(45, 6);
Console.WriteLine("Jugador Σ");
Console.ForegroundColor = ConsoleColor.DarkRed;
Console.SetCursorPosition(45, 7);
Console.WriteLine("Minotauro π");
Console.ForegroundColor = ConsoleColor.Magenta;
Console.SetCursorPosition(45, 8);
Console.WriteLine("Chymera §");
Console.ForegroundColor = ConsoleColor.Blue;
Console.SetCursorPosition(45, 9);
Console.WriteLine("Ogro φ");

```

```

Console.ForegroundColor = ConsoleColor.DarkGray;
Console.SetCursorPosition(45, 10);
Console.WriteLine("Círculo O");
Console.ForegroundColor = ConsoleColor.White;
}

// Permite al usuario seleccionar uno de los 5 mapas
static string MapSelect()
{
    Console.Clear();
    Console.ForegroundColor = ConsoleColor.DarkCyan;
    StreamReader objleer = new StreamReader("SMTitulo.txt");
    string minotauro = objleer.ReadLine();
    while (minotauro != null)
    {
        for (int current = 0; current <= 29; current++)
        {

            Console.WriteLine(minotauro);
            minotauro = objleer.ReadLine();
        }
    }
    objleer.Close();
    Console.ForegroundColor = ConsoleColor.White;

    int mapselec = askint("Ingrese su selección");

    switch (mapselec)
    {
        case 1:
            string opcion = "mapauno.txt";
            return opcion;

        case 2:
            opcion = "mapados.txt";
            return opcion;

        case 3:
            opcion = "mapatres.txt";
            return opcion;
        case 4:
            opcion = "mapacuatro.txt";
            return opcion;

        case 5:
            opcion = "mapacinco.txt";
            return opcion;
        default:
            opcion = "mapauno.txt";
            return opcion;
    }
}

// Permite al usuario entrar al menú de selección de mapas, continuar con los datos de la partida guardada o ver el registro de puntuaciones.
static void modeSelect(ref int[,] matrix, ref int playerx, ref int playery, ref int minox, ref int minoy, ref int chyx, ref int chyy, ref int chybx, ref int chyby, ref int chycx, ref int chycy, ref int ogrx, ref int ogry, ref int ogrbx, ref int ogrby, ref int ogrcx, ref int ogrcy, ref int psyx, ref int psy, ref int psybx, ref int psyby, ref int psycx, ref int psycy, ref int cont, ref int punt, ref char contb, ref int inic, ref int dif, string nombre)
{
    Console.Clear();
    Console.ForegroundColor = ConsoleColor.DarkRed;
    StreamReader objleer = new StreamReader("Smodo.txt");
    string minotauro = objleer.ReadLine();
    while (minotauro != null)
    {
        for (int current = 0; current <= 22; current++)
        {

            Console.WriteLine(minotauro);
            minotauro = objleer.ReadLine();
        }
    }
    objleer.Close();
    Console.ForegroundColor = ConsoleColor.White;
    inic = askint("Seleccione una opción");
    switch (inic)
    {
        case 1:
            string mapa = MapSelect();

```

```

readMaze(mapa, ref matrix);
spawnBlock(matrix, ref minox, ref minoy, ref chyx, ref chyy, ref chybx, ref chyby, ref chycx, ref chycy, ref ogrx, ref ogrby, ref ogrbx, ref ogrcx, ref ogrcy, ref psyx, ref psyb, ref psycx, ref psycy);
difficultySelect(ref matrix, ref playerx, ref playery, ref minox, ref minoy, ref chyx, ref chyy, ref chybx, ref chyby, ref chycx, ref chycy, ref ogrx, ref ogrby, ref ogrbx, ref ogrcx, ref ogrcy, ref psyx, ref psyb, ref psycx, ref psycy);
difficultySelect(ref matrix, ref playerx, ref playery, ref minox, ref minoy, ref chyx, ref chyy, ref chybx, ref chyby, ref chycx, ref chycy, ref ogrx, ref ogrby, ref ogrbx, ref ogrcx, ref ogrcy, ref psyx, ref psyb, ref psycx, ref psycy, ref cont, ref punt, ref dif, nombre);
contb = s_or_n("¿Quieres jugar de nuevo?");
Console.Clear();
break;
case 2:
    readMaze("SavedGame.txt", ref matrix);
    resumePoints(ref punt);
    posResume(ref matrix, ref playerx, ref playery, ref minox, ref minoy, ref chyx, ref chyy, ref chybx, ref chyby, ref chycx, ref chycy, ref ogrx, ref ogrby, ref ogrcx, ref ogrcy, ref psyx, ref psyb, ref psycx, ref psycy);
    difficultySelect(ref matrix, ref playerx, ref playery, ref minox, ref minoy, ref chyx, ref chyy, ref chybx, ref chyby, ref chycx, ref chycy, ref ogrx, ref ogrby, ref ogrbx, ref ogrcx, ref ogrcy, ref psyx, ref psyb, ref psycx, ref psycy, ref cont, ref punt, ref dif, nombre);
    contb = s_or_n("¿Quieres jugar de nuevo ?");
    Console.Clear();
    break;
case 3:
    readPoints();
    contb = s_or_n("¿Quieres volver al menú principal? ");
    break;
}
}

// Permite al usuario seleccionar una dificultad, al hacer esto se decide el número de módulos de control de enemigo que se corren durante la partida.
static void difficultySelect(ref int[,] matrix, ref int playerx, ref int playery, ref int minox, ref int minoy, ref int chyx, ref int chyy, ref int chybx, ref int chyby, ref int chycx, ref int chycy, ref int ogrx, ref int ogrby, ref int ogrbx, ref int ogrcx, ref int ogrcy, ref int psyx, ref int psyy, ref int psyb, ref int psycx, ref int psycy, ref int cont, ref int punt, ref int dif, string nombre)
{
    Console.Clear();
    Console.ForegroundColor = ConsoleColor.DarkCyan;
    StreamReader objleer = new StreamReader("Dif.txt");
    string minotauro = objleer.ReadLine();
    while (minotauro != null)
    {
        Console.ForegroundColor = ConsoleColor.White;
        for (int current = 1; current <= 9; current++)
        {
            Console.WriteLine(minotauro);
            minotauro = objleer.ReadLine();
        }
        Console.ForegroundColor = ConsoleColor.DarkGreen;
        for (int current = 1; current <= 6; current++)
        {
            Console.WriteLine(minotauro);
            minotauro = objleer.ReadLine();
        }
        Console.ForegroundColor = ConsoleColor.DarkYellow;
        for (int current = 1; current <= 6; current++)
        {
            Console.WriteLine(minotauro);
            minotauro = objleer.ReadLine();
        }
        Console.ForegroundColor = ConsoleColor.DarkRed;
        for (int current = 1; current <= 6; current++)
        {
            Console.WriteLine(minotauro);
            minotauro = objleer.ReadLine();
        }
        Console.ForegroundColor = ConsoleColor.White;
    }
    objleer.Close();
    Console.ForegroundColor = ConsoleColor.White;

    dif = askint("Seleccione una dificultad");
    cont = 0;
    switch (dif)
    {
        case 1:
            while (cont == 0)
            {
                printMaze(matrix, punt);

                control(ref playerx, ref playery, ref matrix, ref cont, ref punt, nombre, dif);

                controlMin(ref minox, ref minoy, ref matrix);

```

```

controlChy(ref chyx, ref chyy, ref matrix);
controlOgr(ref ogrx, ref ogyr, ref matrix);
controlPsy(ref psyx, ref psyy, ref matrix);
colisionCheck(matrix, playerx, playery, ref cont);

}

break;

case 2:
while (cont == 0)
{
    printMaze(matrix, punt);

    control(ref playerx, ref playery, ref matrix, ref cont, ref punt, nombre, dif);

    controlMin(ref minox, ref minoy, ref matrix);

    controlChy(ref chyx, ref chyy, ref matrix);
    controlChy(ref chybx, ref chyby, ref matrix);

    controlOgr(ref ogrx, ref ogyr, ref matrix);
    controlOgr(ref ogrbx, ref ogrby, ref matrix);

    controlPsy(ref psyx, ref psyy, ref matrix);
    controlPsy(ref psybx, ref psyby, ref matrix);

    colisionCheck(matrix, playerx, playery, ref cont);

}
break;

case 3:
while (cont == 0)
{
    printMaze(matrix, punt);

    control(ref playerx, ref playery, ref matrix, ref cont, ref punt, nombre, dif);

    controlMin(ref minox, ref minoy, ref matrix);

    controlChy(ref chyx, ref chyy, ref matrix);
    controlChy(ref chybx, ref chyby, ref matrix);
    controlChy(ref chycx, ref chycy, ref matrix);

    controlOgr(ref ogrx, ref ogyr, ref matrix);
    controlOgr(ref ogrbx, ref ogrby, ref matrix);
    controlOgr(ref ogrcx, ref ogrcy, ref matrix);

    controlPsy(ref psyx, ref psyy, ref matrix);
    controlPsy(ref psybx, ref psyby, ref matrix);
    controlPsy(ref psycx, ref psycy, ref matrix);

    colisionCheck(matrix, playerx, playery, ref cont);

}
break;

// Permite al usuario mover al personaje, dentro de los espacios entre paredes. También controla la puntuación del jugador y permite llamar a la función que guarda la partida.
static void control(ref int playerx, ref int playery, ref int[,] matrix, ref int cont, ref int punt, string nombre, int dif)
{
    ConsoleKeyInfo input;
    input = Console.ReadKey(false);

    switch (input.Key)
    {
        case ConsoleKey.RightArrow:
            if (matrix[playerx + 1, playery] == 0 || matrix[playerx + 1, playery] == 3)
            {

                matrix[playerx, playery] = 0;
                playerx++;
                punt++;
                if (matrix[playerx, playery] == 3)

```

```

        {
            punt = punt + 10;
        }
        splayer(playerx, playery, ref matrix);

    }

    if (matrix[playerx + 1, playery] == 4)
    {
        winScreen(ref cont, punt, nombre, dif);
    }
    break;

    case ConsoleKey.LeftArrow:
        if (matrix[playerx - 1, playery] == 0 || matrix[playerx - 1, playery] == 3)
        {
            matrix[playerx, playery] = 0;
            playerx--;
            punt++;
            if (matrix[playerx, playery] == 3)
            {
                punt = punt + 10;
            }
            splayer(playerx, playery, ref matrix);

        }
        if (matrix[playerx - 1, playery] == 4)
        {
            winScreen(ref cont, punt, nombre, dif);
        }
        break;

    case ConsoleKey.DownArrow:
        if (matrix[playerx, playery + 1] == 0 || matrix[playerx, playery + 1] == 3)
        {
            matrix[playerx, playery] = 0;
            playery++;
            punt++;
            if (matrix[playerx, playery] == 3)
            {
                punt = punt + 10;
            }
            splayer(playerx, playery, ref matrix);

        }
        if (matrix[playerx, playery + 1] == 4)
        {
            winScreen(ref cont, punt, nombre, dif);
        }
        break;

    case ConsoleKey.UpArrow:
        if (matrix[playerx, playery - 1] == 0 || matrix[playerx, playery - 1] == 3)
        {
            matrix[playerx, playery] = 0;
            playery--;
            punt++;
            if (matrix[playerx, playery] == 3)
            {
                punt = punt + 10;
            }
            splayer(playerx, playery, ref matrix);

        }
        if (matrix[playerx, playery - 1] == 4)
        {
            winScreen(ref cont, punt, nombre, dif);
        }
        break;

    case ConsoleKey.S:
        saveGame("SavedGame.txt", ref matrix, punt);
        break;
    }

}

// Asigna movimientos aleatorios al minotauro por espacios vacíos
static void controlMin(ref int minox, ref int minoy, ref int[,] matrix)
{
    Random genAleatorio = new Random();
    int randirmin = genAleatorio.Next(1, 5);
}

```

```

switch (randirmin)
{
    case 1:
        if (matrix[minox + 1, minoy] == 0)
        {
            matrix[minox, minoy] = 0;
            minox++;
            smino(minox, minoy, ref matrix);

        }
        break;
    case 2:
        if (matrix[minox - 1, minoy] == 0)
        {
            matrix[minox, minoy] = 0;
            minox--;
            smino(minox, minoy, ref matrix);

        }
        break;
    case 3:
        if (matrix[minox, minoy + 1] == 0)
        {
            matrix[minox, minoy] = 0;
            minoy++;
            smino(minox, minoy, ref matrix);

        }
        break;
    case 4:
        if (matrix[minox, minoy - 1] == 0)
        {
            matrix[minox, minoy] = 0;
            minoy--;
            smino(minox, minoy, ref matrix);

        }
        break;
    }
}

// Asigna movimientos aleatorios a las chimeras por espacios vacíos
static void controlChy(ref int chyx, ref int chyy, ref int[,] matrix)
{
    Random genAleatorio = new Random();
    int randirmin = genAleatorio.Next(1, 5);

    switch (randirmin)
    {
        case 1:
            if (matrix[chyx + 1, chyy] == 0)
            {
                matrix[chyx, chyy] = 0;
                chyx++;
                schy(chyx, chyy, ref matrix);

            }
            break;
        case 2:
            if (matrix[chyx - 1, chyy] == 0)
            {
                matrix[chyx, chyy] = 0;
                chyx--;
                schy(chyx, chyy, ref matrix);

            }
            break;
        case 3:
            if (matrix[chyx, chyy + 1] == 0)
            {
                matrix[chyx, chyy] = 0;
                chyy++;
                schy(chyx, chyy, ref matrix);

            }
            break;
        case 4:
            if (matrix[chyx, chyy - 1] == 0)
            {
                matrix[chyx, chyy] = 0;

```

```

        chyy--;
        schy(chyx, chyy, ref matrix);

    }
    break;

}

}

// Asigna movimientos aleatorios a los ogros por espacios vacíos
static void controlOgr(ref int ogrx, ref int ogry, ref int[,] matrix)
{
    Random genAleatorio = new Random();
    int randirmin = genAleatorio.Next(1, 5);

    switch (randirmin)
    {
        case 1:
            if (matrix[ogrx + 1, ogry] == 0)
            {
                matrix[ogrx, ogry] = 0;
                ogrx++;
                sogr(ogrx, ogry, ref matrix);

            }
            break;
        case 2:
            if (matrix[ogrx - 1, ogry] == 0)
            {
                matrix[ogrx, ogry] = 0;
                ogrx--;
                sogr(ogrx, ogry, ref matrix);

            }
            break;
        case 3:
            if (matrix[ogrx, ogry + 1] == 0)
            {
                matrix[ogrx, ogry] = 0;
                ogry++;
                sogr(ogrx, ogry, ref matrix);

            }
            break;
        case 4:
            if (matrix[ogrx, ogry - 1] == 0)
            {
                matrix[ogrx, ogry] = 0;
                ogry--;
                sogr(ogrx, ogry, ref matrix);

            }
            break;
    }

}

// Asigna movimientos aleatorios a los círculos por espacios vacíos
static void controlPsy(ref int psyx, ref int psyy, ref int[,] matrix)
{
    Random genAleatorio = new Random();
    int randirmin = genAleatorio.Next(1, 5);

    switch (randirmin)
    {
        case 1:
            if (matrix[psyx + 1, psyy] == 0)
            {
                matrix[psyx, psyy] = 0;
                psyx++;
                spsy(psyx, psyy, ref matrix);

            }
            break;
        case 2:
            if (matrix[psyx - 1, psyy] == 0)
            {
                matrix[psyx, psyy] = 0;
                psyx--;
                spsy(psyx, psyy, ref matrix);

            }
            break;
    }

}

```

```

    spsy(psyx, psyy, ref matrix);

}

break;
case 3:
if (matrix[psyx, psyy + 1] == 0)
{
    matrix[psyx, psyy] = 0;
    psyy++;
    spsy(psyx, psyy, ref matrix);

}
break;
case 4:
if (matrix[psyx, psyy - 1] == 0)
{
    matrix[psyx, psyy] = 0;
    psyy--;
    spsy(psyx, psyy, ref matrix);

}
break;

}

```

// Métodos que escriben el valor de cada personaje dentro del arreglo bidimensional después de cada movimiento

```

static void splayr(int playerx, int playery, ref int[,] matrix)
{
    matrix[playerx, playery] = 5;
}
static void smino(int minox, int minoy, ref int[,] matrix)
{
    matrix[minox, minoy] = 6;
}
static void schy(int chyx, int chyy, ref int[,] matrix)
{
    matrix[chyx, chyy] = 7;
}
static void sogr(int ogrx, int ogry, ref int[,] matrix)
{
    matrix[ogrx, ogry] = 8;
}
static void spsy(int psyx, int psyy, ref int[,] matrix)
{
    matrix[psyx, psyy] = 9;
}

```

// Despliega la pantalla de victoria cuando el jugador completa un laberinto, finaliza la partida y guarda los datos del jugador y la partida en el archivo de puntuaciones

```
StreamReader objleer = new StreamReader("MinoArtLC.txt");
string minotauro = objleer.ReadLine();
while (minotauro != null)
{
    for (int current = 0; current <= 31; current++)
    {
        Console.WriteLine(minotauro);
        minotauro = objleer.ReadLine();
    }
    objleer.Close();
    Console.ForegroundColor = ConsoleColor.White;
    Console.WriteLine("");
    Console.WriteLine("Puntuación: {0}", punt);
    cont++;
}
```



```

        string value = Convert.ToString(matrix[j, i]);
        objgrabar.WriteLine("{0}, " + value);
    }
    objgrabar.WriteLine("\n");
}
objgrabar.Close();

File.WriteAllText(@"ScoreSave.txt", string.Empty);
StreamWriter objgrabarb = new StreamWriter("ScoreSave.txt", true);
objgrabarb.Write(punt);
objgrabarb.Close();
}

// Escribe los datos del jugador ganador de partida y los agrega al archivo de registro de puntajes.
static void writeScore(string nombre, int punt, string dif)
{
    StreamWriter objgrabar = new StreamWriter("Puntaje.txt", true);
    objgrabar.WriteLine("{0}\t{1}\t{2}", nombre, punt, dif);
    objgrabar.WriteLine("\n");
    objgrabar.Close();
}

// Usa la información de la partida guardada para actualizar los valores de posición de los personajes antes de que se reanude la partida guardada.
static void posResume(ref int[,] arr, ref int playerx, ref int playery, ref int minox, ref int minoy, ref int chyx, ref int chyy, ref int chybx, ref int chyby, ref int chycx, ref int chycy, ref int ogrx, ref int ogry, ref int ogrbx, ref int ogrby, ref int ogrcx, ref int ogrcy, ref int psyx, ref int psybx, ref int psby, ref int psycx, ref int psycy)
{
    Console.Clear();
    for (int j = 0; j < arr.GetLength(1); j++)
    {
        for (int i = 0; i < arr.GetLength(0); i++)
        {
            if (arr[i, j] == 5)
            {
                playerx = i;
                playery = j;
                arr[i, j] = 0;
            }
            if (arr[i, j] == 6)
            {
                minox = i;
                minoy = j;
                arr[i, j] = 0;
            }
            if (arr[i, j] == 7)
            {
                chyx = i;
                chyy = j;
                arr[i, j] = 0;
            }
            if (arr[i, j] == 7)
            {
                chybx = i;
                chyby = j;
                arr[i, j] = 0;
            }
            if (arr[i, j] == 7)
            {
                chycx = i;
                chycy = j;
                arr[i, j] = 0;
            }
            if (arr[i, j] == 8)
            {
                ogrx = i;
                ogry = j;
                arr[i, j] = 0;
            }
            if (arr[i, j] == 8)
            {
                ogrbx = i;
                ogrby = j;
                arr[i, j] = 0;
            }
        }
    }
}

```

```

        if (arr[i, j] == 8)
    {
        ogrcx = i;
        ogrcy = j;
        arr[i, j] = 0;
    }

        if (arr[i, j] == 9)
    {
        psyx = i;
        psyv = j;
        arr[i, j] = 0;
    }
        if (arr[i, j] == 9)
    {
        psybx = i;
        psyby = j;
        arr[i, j] = 0;
    }
        if (arr[i, j] == 9)
    {
        psycx = i;
        psycy = j;
        arr[i, j] = 0;
    }
    }

    Console.WriteLine("\n");
}

ConsoleCursorPosition(0, 1);
}

//Actualiza la variable de puntuación con el valor en el archivo de puntuación de la partida guardada
static void resumePoints(ref int punt)
{
    StreamReader objleerb = new StreamReader("ScoreSave.txt");
    punt = Convert.ToInt32(objleerb.ReadLine());
    objleerb.Close();
}

// Pide un valor entero al usuario
static int askint(string mes)
{
    int output;
    Console.WriteLine("{0}", mes);
    output = Convert.ToInt32(Console.ReadLine());
    return output;
}

// Checa la posición del jugador con respecto a los enemigos y termina la partida en caso de estar en contacto con ellos
static void colisionCheck(int[,] matrix, int playerx, int playery, ref int cont)
{
    if (matrix[playerx + 1, playery] == 6 || matrix[playerx + 1, playery] == 7 || matrix[playerx + 1, playery] == 8 || matrix[playerx + 1, playery] == 9)
    {
        gameOver(ref cont);
    }
    if (matrix[playerx - 1, playery] == 6 || matrix[playerx - 1, playery] == 7 || matrix[playerx - 1, playery] == 8 || matrix[playerx - 1, playery] == 9)
    {
        gameOver(ref cont);
    }
    if (matrix[playerx, playery + 1] == 6 || matrix[playerx, playery + 1] == 7 || matrix[playerx, playery + 1] == 8 || matrix[playerx, playery + 1] == 9)
    {
        gameOver(ref cont);
    }
    if (matrix[playerx, playery - 1] == 6 || matrix[playerx, playery - 1] == 7 || matrix[playerx, playery - 1] == 8 || matrix[playerx, playery - 1] == 9)
    {
        gameOver(ref cont);
    }
}

// Pregunta al usuario si quiere regresar al menú principal o finalizar el programa
static char s_o_n(string mes)
{
    Console.WriteLine("");
    Console.WriteLine("{0} (s/n) <enter>", mes);
    char output = Convert.ToChar(Console.ReadLine());
}

```

```

Console.Clear();
return output;
}

// Pide una cadena de caracteres al usuario
static string askstr(string mes)
{
    string output;
    Console.WriteLine("{0}", mes);
    output = Console.ReadLine();
    return output;
}

//Forza que los enemigos aparezcan sólamente en los espacios vacíos del laberinto
static void spawnBlock(int[,] arr, ref int minox, ref int minoy, ref int chyx, ref int chyy, ref int chybx, ref int chybx, ref int chycx, ref int chycy, ref int ogrx, ref int ogry, ref int ogrbx, ref int ogrby, ref int ogrcx, ref int ogrcy, ref int psyx, ref int psyy, ref int psybx, ref int psyby, ref int psycx, ref int psycy)
{
    Random genSpawn = new Random();

    if (arr[minox, minoy] != 0)
    {
        minox = genSpawn.Next(1, 40);
        minoy = genSpawn.Next(1, 20);
    }

    if (arr[chyx, chyy] != 0)
    {
        chyx = genSpawn.Next(1, 40);
        chyy = genSpawn.Next(1, 20);
    }

    if (arr[chybx, chyby] != 0)
    {
        chybx = genSpawn.Next(1, 40);
        chyby = genSpawn.Next(1, 20);
    }

    if (arr[chycx, chycy] != 0)
    {
        chycx = genSpawn.Next(1, 40);
        chycy = genSpawn.Next(1, 20);
    }

    if (arr[ogrx, ogry] != 0)
    {
        ogrx = genSpawn.Next(1, 40);
        ogry = genSpawn.Next(1, 20);
    }

    if (arr[ogrbx, ogrby] != 0)
    {
        ogrbx = genSpawn.Next(1, 40);
        ogrby = genSpawn.Next(1, 20);
    }

    if (arr[ogr cx, ogr cy] != 0)
    {
        ogrcx = genSpawn.Next(1, 40);
        ogrcy = genSpawn.Next(1, 20);
    }

    if (arr[psyx, psyy] != 0)
    {
        psyx = genSpawn.Next(2, 39);
        psyy = genSpawn.Next(2, 19);
    }

    if (arr[psybx, psyby] != 0)
    {
        psybx = genSpawn.Next(2, 39);
        psyby = genSpawn.Next(2, 19);
    }
}

```

```
    if (arr[psyCx, psyCy] != 0)
    {
        psyCx = genSpawn.Next(2, 39);
        psyCy = genSpawn.Next(2, 19);
    }

}
```