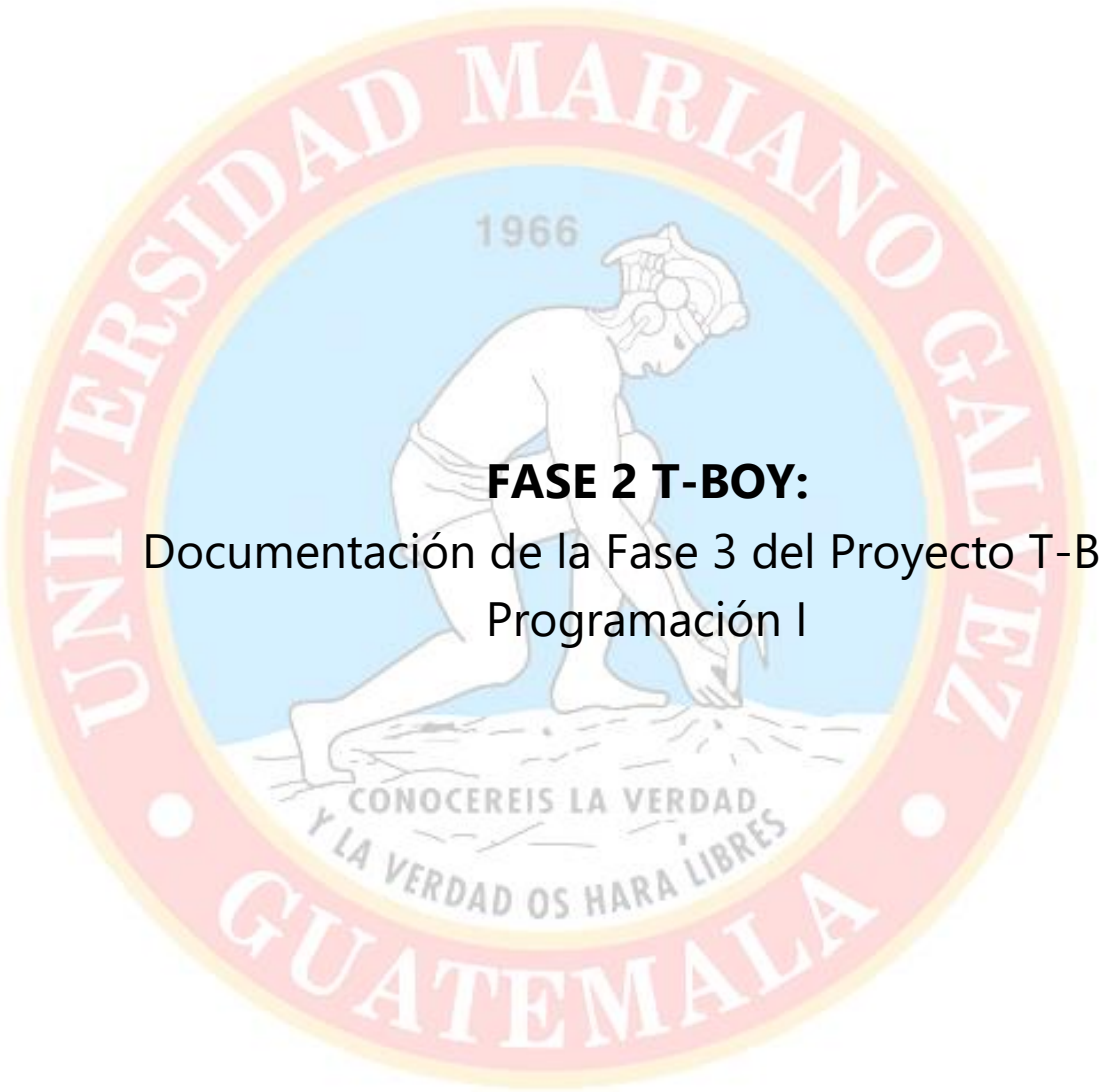
UNIVERSIDAD MARIANO GALVEZ DE GUATEMALA

CENTRO UNIVERSITARIO DE MAZATENANGO - SUCHITEPEQUEZ
FACULTAD DE INGENIERIA EN SISTEMAS
ING. AXEL AGUILAR
CURSO PROGRAMACION I

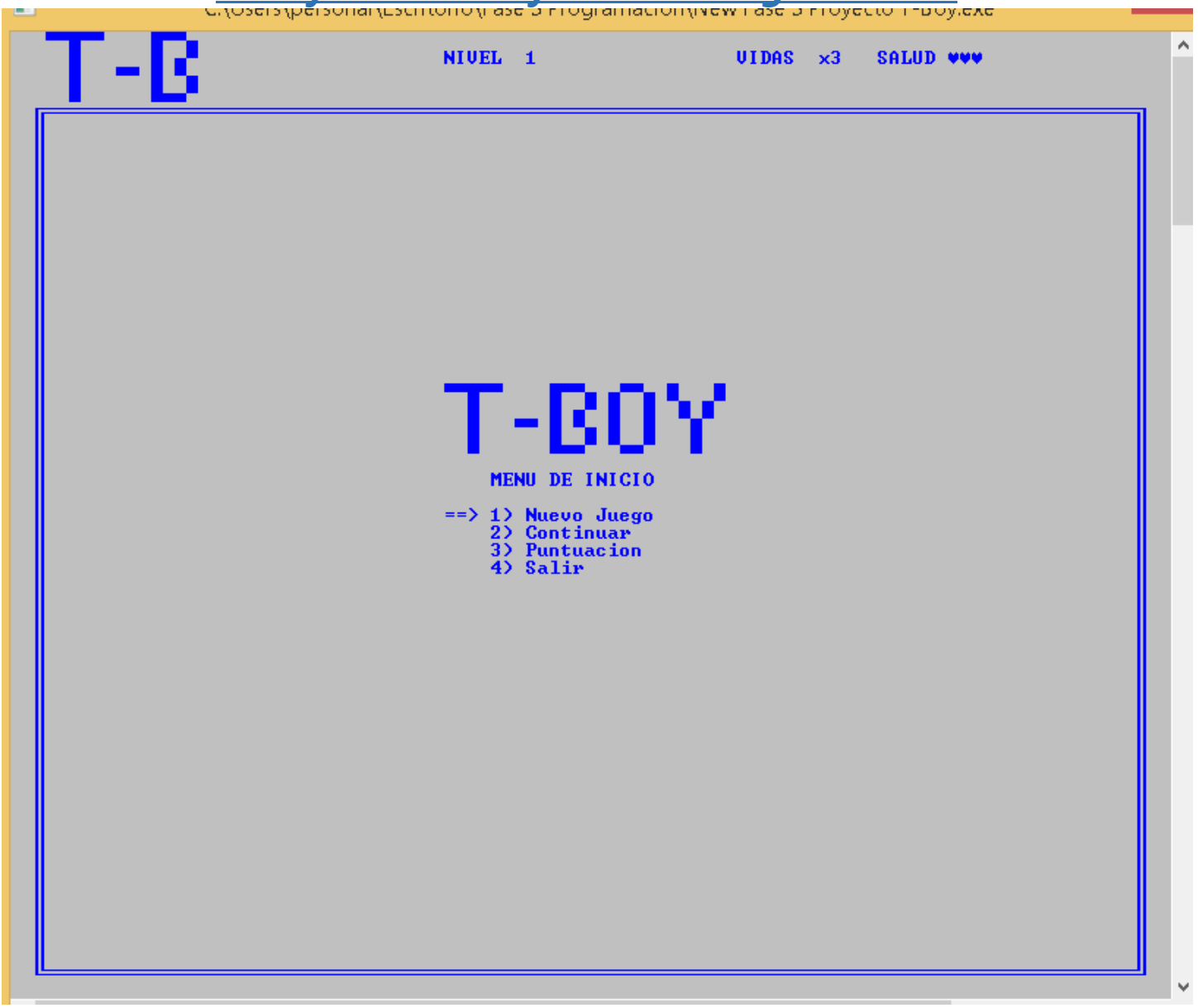


FASE 2 T-BOY:

Documentación de la Fase 3 del Proyecto T-Boy
Programación I

OSCAR RODOLFO CHÁVEZ CAMEY
3090-19-13543

T-Boy Fase 3 Proyecto de Programación



Proyecto del 3er Ciclo de Ingeniería en Sistemas y Ciencias de la Computación en el Curso de Programación 1.

La Fase 3, como fase de presentación agrega las ultimas opciones y correcciones al juego de T-Boy, como ultima característica el poder crear un nombre usuario poder cargarlo e incluso ver las puntuaciones de otros usuarios.

A continuación, presentare el código fuente con algunas anotaciones y describiré lo que se está ejecutando en ese fragmento de código.

.

```
1   #include <iostream>
2   #include <conio.h>
3   #include <stdio.h>
4   #include <windows.h>
5   #include <stdlib.h>
6   #include <time.h>
7   #include <fstream>
8   #include <cstring>
9   using namespace std;
10  #define ARRIBA 72
11  #define IZQUIERDA 75
12  #define DERECHA 77
13  #define ABAJO 80
14  #define ENTER 13
15  #define SALTOD 'd'
16  #define SALTOI 'a'
17  #define BUSCAR 's'
18
```

Librerías

En esta parte del código se declaran las funciones y variables presentes en la ejecución del programa como de las librerías que hacen posible la ejecución del código fuente; de igual forma en este espacio están definidas las teclas de movimiento del jugador, la tecla "d" realiza un salto hacia la derecha, la tecla "a" realiza un salto hacia la izquierda y la tecla "s" se utiliza para ingresar a la puerta secreta del final del juego únicamente si el jugador está en la posición correcta.

VARIABLES GLOBALES

En este segmento de código inician las variables globales, específicamente muestra el cuerpo del personaje y la explosión de este al chocar contra un barril enemigo o caer de un piso demasiado alto, junto con la línea de borrado de personaje la cual se utiliza para la transición de movimiento del personaje

```
19  //VARIABLES GLOBALES
20                      //VARIABLES JUGADOR 1
21  char jugador1_1[]={' ','@',' ',0};
22  char jugador1_2[]={'#','H','#',0};
23  char jugador1_3[]={' ','H',' ',0};
24  char jugador1_4[]={'#',' ','#',0};
25
26  char explocion_l1[]={' ','x',' ',0};
27  char explocion_l2[]={'x','x','x',0};
28  char explocion_l3[]={'X','x','X',0};
29  char explocion_l4[]={' ','x',' ',0};
30
31  char explocion_r1[]={'x',' ','x',0};
32  char explocion_r2[]={' ',' ',' ',0};
33  char explocion_r3[]={' ',' ',' ',0};
34  char explocion_r4[]={'x',' ','x',0};
35
36  char explocion_s1[]={' ',' ',' ',0};
37  char explocion_s2[]={' ',' ',' ',0};
38  char explocion_s3[]={' ','x',' ',0};
39  char explocion_s4[]={'x','x','x',0};
40
```

```
43                      //VARIABLES ENEMIGOS
44  char enemigo_l1[]={' ','@',' ',0};
45  char enemigo_l2[]={'@','@','@',0};
46  char enemigo_l3[]={' ','@',' ',0};
47
48  char romper_enemigo_l1[]={' ','#',' ',0};
49  char romper_enemigo_l2[]={'#','#','#',0};
50  char romper_enemigo_l3[]={' ','#',' ',0};
51
52  char romper_enemigo_r1[]={'#',' ','#',0};
53  char romper_enemigo_r2[]={' ',' ',' ',0};
54  char romper_enemigo_r3[]={'#',' ','#',0};
55
56  char eliminar_enemigo[]={' ',' ',' ',0};
```

Variables enemigo barril

En estas variables se encuentra el cuerpo del enemigo barril junto a las variables para la explosión al momento de chocar contra el jugador o al terminar su recorrido, por último, la variable eliminar que se utiliza para la transición del movimiento del enemigo.

```
58          //VARIABLES GENERALES
59  int Num_vidas_p1=3;
60  int Corazones_p1=3;
61  int puntuacion_p1=0;
62  int x_p1=9;
63  int y_p1=49;
64  bool Activo=false;
65  int f=10;
66  bool saltod_on=false;
67  bool saltoi_on=false;
68  int cs1=0,cs2=0,cs3=0;
69  bool unico=false;
70
71  int cc=1,xr,yr;
72
73  int x_b1=65,y_b1=10;
74  int x_b2=65,y_b2=10;
75  int x_b3=65,y_b3=10;
76  int x_b4=65,y_b4=10;
77  int x_b5=65,y_b5=10;
78  int t=0;
```

Variables Generales

Dentro de estas variables se encuentra la vida y salud, puntuación, ubicación, saltos y caída del jugador 1, así como las ubicaciones de los barriles y la velocidad de los mismos.

```
80  int x_mm=81,y_mm=5;
81  bool mover_mm=false;
82  bool mover_mm2=false;
83
84  int x_rabbit=67;
85  int y_rabbit=6;
86
87  bool finA=false;
88  bool finB=false;
89  bool finC=false;
90  bool finD=false;
91  bool finE=false;
92  bool inicioA=false;
93  bool inicioB=false;
94  bool inicioC=false;
95  bool inicioD=false;
96  bool inicioE=false;
97  bool dw1=true;
98  bool dg1=true;
99  bool dw2=true;
100 bool dg2=true;
101 bool dw3=true;
102 bool dg3=true;
```

Variables Generales

Continuando con las variables generales en este segmento se encuentran las variables de ubicación del monstruo junto a las condiciones para su movimiento, la ubicación del conejo, como los indicadores de inicio y fin del recorrido de cada barril junto a la activación para bajar de un piso a otro.

```
//FUNCIONES GLOBALES
void gotoxy(int x,int y);
void mover_enemigo1();
void pintar_plataforma();
void mostrar_escalera();
void mostrar_salud();
void Score(int x,int y);
int prototipo(int &x, int &y, bool &dw, bool
int impacto(int &x,int &y,bool &dw,bool &dg);

struct Partida{              //ESTRUTURA DE DA
    char nombre[20];
    int puntaje;
    int vidas;
    int salud;
}partidas[2];

//FUNCIONES ALAMACENAMIENTO DE DATOS
int eleccion(const char titulo[], const char
void mennu();
void Nueva_partida();
void Newprofile();
void Continuar();
```

Funciones y Variables de Almacenamiento

En este segmento se muestra las funciones globales que son utilizadas en distintas ubicaciones de las direcciones del programa. Justo debajo esta la declaración de la estructura de datos para almacenar la partida del jugador y algunas de las funciones del menú de Inicio, dicho menú se utiliza para cargar y guardar los datos de usuario del jugador.

```
void escritura(int x,int ubicacion,
void cargardatos(int x,int ubicacion
void Puntuaciones();
void limpiar(int x,int y);
int slot=0;
bool nuevaentrada=false;
bool startg=true;
bool salir_del_juego=false;

bool game_over=false;
bool game_start=false;
bool level_complete=false;
int Nivel=1;
int j=1;
```

Funciones Generales

Continuando con las funciones para almacenar los datos de usuario del jugador y datos booleanos para la correcta ejecución del código.

```
void pintar_rabbit()
{
    gotoxy(x_rabbit,y_rabbit);  printf("X  X");
    gotoxy(x_rabbit,y_rabbit+1);printf("(..)");
    gotoxy(x_rabbit,y_rabbit+2);printf("()()");
}
void borrar_rabbit()
{
    gotoxy(x_rabbit,y_rabbit);  printf("    ");
    gotoxy(x_rabbit,y_rabbit+1);printf("    ");
    gotoxy(x_rabbit,y_rabbit+2);printf("    ");
}
```

Funciones del Conejo

Funciones para pintar y borrar lo cual hace posible la animación de movimiento del mismo.

```
160  void pintar_jugador1(void){
161          gotoxy(x_p1,y_p1);puts(jugador1_1);
162          gotoxy(x_p1,y_p1+1);puts(jugador1_2);
163          gotoxy(x_p1,y_p1+2);puts(jugador1_3);
164          gotoxy(x_p1,y_p1+3);puts(jugador1_4);
165  }
166  void eliminar_jugador(void){
167          gotoxy(x_p1,y_p1);puts(borrar_jugador1);
168          gotoxy(x_p1,y_p1+1);puts(borrar_jugador1);
169          gotoxy(x_p1,y_p1+2);puts(borrar_jugador1);
170          gotoxy(x_p1,y_p1+3);puts(borrar_jugador1);
171  }
```

Funciones pintar y borrar jugador 1

Función para pintar y borrar el cuerpo del jugador o protagonista utilizado tanto para ubicarlo en sus coordenadas como para animar el movimiento del jugador.

```
172  void Explotar()
173  {
174          gotoxy(x_p1,y_p1);puts(explocion_l1);
175          gotoxy(x_p1,y_p1+1);puts(explocion_l2);
176          gotoxy(x_p1,y_p1+2);puts(explocion_l3);
177          gotoxy(x_p1,y_p1+3);puts(explocion_l4);
178          Sleep(60);
179
180          gotoxy(x_p1,y_p1);puts(explocion_r1);
181          gotoxy(x_p1,y_p1+1);puts(explocion_r2);
182          gotoxy(x_p1,y_p1+2);puts(explocion_r3);
183          gotoxy(x_p1,y_p1+3);puts(explocion_r4);
184          Sleep(60);
185
186          gotoxy(x_p1,y_p1);puts(explocion_s1);
187          gotoxy(x_p1,y_p1+1);puts(explocion_s2);
188          gotoxy(x_p1,y_p1+2);puts(explocion_s3);
189          gotoxy(x_p1,y_p1+3);puts(explocion_s4);
190          Sleep(60);
191
192          pintar_jugador1();
193          f=10;
194  }
```

Función Explotar

Función para animar el choque del personaje con un enemigo o cuando el personaje sufra daño por caída.

Funciones pintar, borrar y choque enemigo

Funciones para mostrar la ubicación del barril enemigo, para borrar cada vez que este avance, utilizada también para la animación del movimiento, caída y función de choque que es la animación cuando el barril choca contra el jugador o cuando termina el recorrido en pantalla.

```
195  void pintar_enemigo3(int x, int y){
196          gotoxy(x,y);  puts(enemigo_l1);
197          gotoxy(x,y+1);puts(enemigo_l2);
198          gotoxy(x,y+2);puts(enemigo_l3);
199  }
200  void borrar_enemigo3(int x, int y){
201          gotoxy(x,y);  puts(eliminar_enemigo);
202          gotoxy(x,y+1);puts(eliminar_enemigo);
203          gotoxy(x,y+2);puts(eliminar_enemigo);
204  }
205  void choque_enemigo3(int x,int y,bool &dw,bool &dg)
206  {
207          gotoxy(x,y);  puts(romper_enemigo_l1);
208          gotoxy(x,y+1);puts(romper_enemigo_l2);
209          gotoxy(x,y+2);puts(romper_enemigo_l3);
210          Sleep(60);
211
212          gotoxy(x,y);  puts(romper_enemigo_r1);
213          gotoxy(x,y+1);puts(romper_enemigo_r2);
214          gotoxy(x,y+2);puts(romper_enemigo_r3);
215          Sleep(60);
216
```

```
220  void mostrar_salud(){
221      gotoxy(62,1);printf("VIDAS  x%d",Num_vidas_p1);
222
223      gotoxy(74,1);printf("SALUD");
224      gotoxy(80,1);printf("     ");
225      for(int i=0;i<Corazones_p1;i++){
226          gotoxy(80+i,1);printf("%c",3);
227      }
228      gotoxy(37,1);printf("NIVEL  %d",Nivel);
229
230      if(startg==true && salir_del_juego==false)
231      {
232      }else{
233      ifstream fentrada("caset.bin",ios::in | ios::bin
234      fentrada.read(reinterpret_cast<char *>(&partidas
235      gotoxy(37,3);cout<<partidas[slot].nombre;
236      fentrada.close();
237      }
238  }
```

Función mostrar _salud

La función pintar mostrar salud ubica la cantidad de vidas y salud que tiene el jugador 1 en la parte superior derecha de la pantalla además del mostrar el nombre del usuario y el nivel en que se encuentra.

Función pintar_monstruo y pintar_monstruo2

Existen dos funciones para pintar al monstruo ya que este se mueve en dirección izquierda o derecha, de esta forma se logra que parezca que este se mueve en el último piso.

```
239  void pintar_monstruo()
240  {
241      gotoxy(x_mm,y_mm);  printf("%c%c%c%c%c%c%c", 32,219,219,21
242      gotoxy(x_mm,y_mm+1);printf("%c%c%c%c%c%c%c",219,219,219,21
243      gotoxy(x_mm,y_mm+2);printf("%c%c%c%c%c%c%c", 32,219,219,21
244      gotoxy(x_mm,y_mm+3);printf("%c%c%c%c%c%c%c", 32, 32,219,21
245      gotoxy(x_mm,y_mm+4);printf("%c%c%c%c%c%c%c",219,219,219,21
246      gotoxy(x_mm,y_mm+5);printf("%c%c%c%c%c%c%c", 32,219,219,21
247      gotoxy(x_mm,y_mm+6);printf("%c%c%c%c%c%c%c", 32,219,219,21
248      gotoxy(x_mm,y_mm+7);printf("%c%c%c%c%c%c%c",219,219,219,21
249  }
250  void pintar_monstruo2()
251  {
252      gotoxy(x_mm,y_mm);  printf("%c%c%c%c%c%c%c", 32, 32,219,21
253      gotoxy(x_mm,y_mm+1);printf("%c%c%c%c%c%c%c", 32,219,219,21
254      gotoxy(x_mm,y_mm+2);printf("%c%c%c%c%c%c%c", 32,219,219,21
255      gotoxy(x_mm,y_mm+3);printf("%c%c%c%c%c%c%c", 32, 32,219,21
256      gotoxy(x_mm,y_mm+4);printf("%c%c%c%c%c%c%c",219, 32, 32,21
257      gotoxy(x_mm,y_mm+5);printf("%c%c%c%c%c%c%c",219, 32, 32,21
258      gotoxy(x_mm,y_mm+6);printf("%c%c%c%c%c%c%c", 32,219, 32,21
259      gotoxy(x_mm,y_mm+7);printf("%c%c%c%c%c%c%c", 32, 32,219,21
260  }
```

```
261  void borrar_monstruo()
262  {
263      gotoxy(x_mm,y_mm);  printf("       ");
264      gotoxy(x_mm,y_mm+1);printf("       ");
265      gotoxy(x_mm,y_mm+2);printf("       ");
266      gotoxy(x_mm,y_mm+3);printf("       ");
267      gotoxy(x_mm,y_mm+4);printf("       ");
268      gotoxy(x_mm,y_mm+5);printf("       ");
269      gotoxy(x_mm,y_mm+6);printf("       ");
270      gotoxy(x_mm,y_mm+7);printf("       ");
271  }
272  void explotar_monstruo()
273  {
274          gotoxy(x_mm,y_mm);  printf("   +   ");
275          gotoxy(x_mm,y_mm+1);printf("   +   ");
276          gotoxy(x_mm,y_mm+2);printf("  +++  ");
277          gotoxy(x_mm,y_mm+3);printf(" +++++ ");
278          gotoxy(x_mm,y_mm+4);printf(" +++++ ");
279          gotoxy(x_mm,y_mm+5);printf("  +++  ");
280          gotoxy(x_mm,y_mm+6);printf("   +   ");
281          gotoxy(x_mm,y_mm+7);printf("   +   ");
282          Sleep(80);
283
```

Función borrar_monstruo y explotar_monstruo

La primera función borra la silueta en la ubicación del enemigo para que este pueda moverse por la pantalla sin dejar rastro y la función explotar_monstruo muestra una pequeña animación de estallido cuando el jugador completa el nivel 3.

Función gotoxy

Función utilizada para ubicar los diferentes elementos del juego en coordenadas bidimensionales.

```
305  void gotoxy(int x,int y){
306      HANDLE hcon;
307      hcon = GetStdHandle(STD_OUTPUT_HANDLE);
308      COORD dwPos;
309      dwPos.X=x;
310      dwPos.Y=y;
311      SetConsoleCursorPosition(hcon,dwPos);
312  }
```

Función ocultarcursor

Función utilizada para desaparecer el puntero de escritura de la ejecución del juego.

```
313  void ocultarcursor(){
314      HANDLE hCon;
315      hCon = GetStdHandle(STD_OUTPUT_HANDLE);
316      CONSOLE_CURSOR_INFO cci;
317      cci.dwSize = 100;
318      cci.bVisible = FALSE;
319      SetConsoleCursorInfo(hCon,&cci);
320  }
```

```
321  void escenario(){
322      for(int i=3;i<=96;i++)
323      {
324          gotoxy(i,4);printf("%c",20
325          gotoxy(i,53);printf("%c",2
326      }
327      for(int i=5;i<=52;i++)
328      {
329          gotoxy(2,i);printf("%c",18
330          gotoxy(96,i);printf("%c",1
331      }
332      gotoxy( 2, 4);printf("%c",201)
333      gotoxy(96, 4);printf("%c",187)
334      gotoxy( 2,53);printf("%c",200)
335      gotoxy(96,53);printf("%c",188)
336
337      gotoxy(3,0);printf("%c%c%c%c%c
338      gotoxy(3,1);printf("%c%c%c%c%c
339      gotoxy(3,2);printf("%c%c%c%c%c
340      gotoxy(3,3);printf("%c%c%c%c%c
341
342      mostrar_salud();
343  }
```

Función escenario

Función utilizada para dibujar el entorno de juego, pinta limites, título y salud en pantalla.

Función findeljuego, cubocuadrado y bloqueplataforma

La función fin del juego dibuja la pantalla de salida o terminación del juego, la función cubo cuadrado dibuja el cuadrado buff donde el personaje puede subir y bajar, y la función bloque plataforma dibuja plataforma en las coordenadas que se le asignen.

```
344  void findeljuego(){
345      gotoxy(30,25);printf("%c%c%c
346      gotoxy(30,26);printf("%c%c%c
347      gotoxy(30,27);printf("%c%c%c
348      gotoxy(30,28);printf("%c%c%c
349      gotoxy(30,29);printf("%c%c%c
350  }
351  void cubocuadrado(int x,int y)
352  {
353      gotoxy(x,y)  ;printf("%c%c%c%
354      gotoxy(x,y+1);printf("%c%c%c%
355      gotoxy(x,y+2);printf("%c%c%c%
356  }
357  void bloqueplataforma(int x,int y
358  {
359      gotoxy(x,y);      printf("%c",
360      gotoxy(x+74,y);   printf("%c",
361      gotoxy(x,y+1);    printf("%c",
362      gotoxy(x+74,y+1);printf("%c",
363
364      for(int i=x+1;i<=x+73;i++){
```

```
369  void pintar_plataforma(){
370      escenario();
371      bloqueplataforma( 3,43);
372      bloqueplataforma(21,33);
373      bloqueplataforma( 3,23);
374      bloqueplataforma(21,13);
375      cubocuadrado(92,50);
376      cubocuadrado( 3,40);
377      cubocuadrado(92,30);
378      cubocuadrado( 3,20);
379      cubocuadrado(92,10);
380      cubocuadrado(88,10);
381      cubocuadrado(90, 7);
382      for(int i=55;i<=70;i++){
383          gotoxy(i,9);  printf("%c",2
384      pintar_rabbit();
385      gotoxy( 3,45);printf("%c%c%c%c"
386      gotoxy( 3,46);printf("%c%c%c%c"
387      gotoxy( 3,47);printf("%c%c%c%c"
388      gotoxy( 3,48);printf("%c%c%c%c"
389      gotoxy( 3,49);printf("%c%c%c%c"
```

Función pintar plataforma

Esta función carga los bloques buff, plataformas, pinta la posición inicial del conejo y su plataforma, el bloque inicial de carrera del jugador y la posición inicial del monstruo.

Función bloqueescalera y mostrar_escalera

Función para pintar un bloque de escalera en cualquier parte del juego según las coordenadas ingresadas, la función mostrar_escalera es la encargada de pintar las escaleras que conectan cada nivel de juego.

```
395  void bloqueescalera(int x, int y)
396  {
397      gotoxy(x,y);printf("%c%c%c%c%c",195,196,196,196,
398      gotoxy(x,y+1);printf("%c%c%c%c%c",195,196,196,19
399      gotoxy(x,y+2);printf("%c%c%c%c%c",195,196,196,19
400      gotoxy(x,y+3);printf("%c%c%c%c%c",195,196,196,19
401      gotoxy(x,y+4);printf("%c%c%c%c%c",195,196,196,19
402      gotoxy(x,y+5);printf("%c%c%c%c%c",195,196,196,19
403      gotoxy(x,y+6);printf("%c%c%c%c%c",195,196,196,19
404      gotoxy(x,y+7);printf("%c%c%c%c%c",195,196,196,19
405      gotoxy(x,y+8);printf("%c%c%c%c%c",195,196,196,19
406  }
407  void mostrar_escalera(){
408
409      bloqueescalera(70,44);          //INSTANCIA DE
410      bloqueescalera(24,34);          //INSTANCIA DE
411      bloqueescalera(70,24);          //INSTANCIA DE
412      bloqueescalera(24,14);          //INSTANCIA DE
413
414      gotoxy(50, 9);printf("%c%c%c%c%c",195,196,196,19
415      gotoxy(50,10);printf("%c%c%c%c%c",195,196,196,19
416      gotoxy(50,11);printf("%c%c%c%c%c",195,196,196,19
417      gotoxy(50,12);printf("%c%c%c%c%c",195,196,196,19
```

```
437  void saltar_derecha()
438  {
439      if(cs1<3 && saltod_on==true)
440      {
441          eliminar_jugador();
442          x_p1++;y_p1--;cs1++;
443      }
444      if((cs2>=3 && cs2<7) && saltod_on==true)
445      {
446          eliminar_jugador();
447          x_p1++;cs1++;
448          if(
449                  ((x_p1>=88 && x_p1<=95) && cs
450             )
451          {
452              saltod_on=false;
453              cs1=0,cs2=0,cs3=0;
454          }
455      }
456      if((cs3>=7 && cs3<=9) && saltod_on==true)
457      {
458          eliminar_jugador();
459          x_p1++;y_p1++;cs1++;
```

Función saltar_derecha

Esta función opera el salto del jugador en dirección derecha en determinada posición, debe cumplirse una serie de coordenadas para acceder a la animación y desplazamiento de salto.

Función saltar_izquierda

Esta función opera el salto del jugador en dirección izquierda en determinada posición, debe cumplirse una serie de coordenadas para acceder a la animación y desplazamiento de salto.

```
480  void saltar_izquierda()
481  {
482      if(cs1<3 && saltoi_on==true)
483      {
484          eliminar_jugador();
485          x_p1--;y_p1--;cs1++;
486      }
487      if((cs2>=3 && cs2<7) && saltoi_on==true)
488      {
489          eliminar_jugador();
490          x_p1--;cs1++;
491          if(
492                  ((x_p1>= 1 && x_p1<= 8) && cs
493             )
494          {
495              saltoi_on=false;
496              cs1=0,cs2=0,cs3=0;
497          }
498      }
499      if((cs3>=7 && cs3<=9) && saltoi_on==true)
500      {
501          eliminar_jugador();
502          x_p1--;y_p1++;cs1++;
```

```
523 void mover_monstruo()
524 {
525     if(mover_mm==true && x_mm>72)
526     {
527         borrar_monstruo();
528         x_mm--;
529         pintar_monstruo();
530     }
531     if(mover_mm2==true && x_mm<81)
532     {
533         borrar_monstruo();
534         x_mm++;
535         pintar_monstruo2();
536     }
537 }
538 void caer()
539 {
540     eliminar_jugador();
541     y_p1++;
542     pintar_jugador1();
543 }
```

Función mover_monstruo

Genera animación del movimiento del monstruo cada vez que el barril descienda entre plataformas.

Función caer

Desciende al personaje, independientemente de la coordenada en la que se encuentre.

Función muerte_caida

Detecta el momento cuando el jugador se encuentra fuera de la plataforma y anima la caída al nivel inferior restándole una vida, en caso el personaje tenga media salud esta será restaurada al restar esa vida.

```
544 void muerte_caida()
545 {
546     if(     ((x_p1>=78 && x_p1<=93) && (y_p1==39))
547             ((x_p1>= 3 && x_p1<=17) && (y_p1==29))
548             ((x_p1>=78 && x_p1<=93) && (y_p1==19))
549             ((x_p1>= 3 && x_p1<=17) && (y_p1== 9))
550       ){
551             Activo=true;
552       }
553       if(f==0 && Activo==true)
554             {
555                 Activo=false;
556                 Explotar();
557                 Num_vidas_p1--;
558                 Corazones_p1=3;
559                 mostrar_salud();
560             }
561       if(Activo==true)
562             {
563                 f--;
564                 caer();
565             }
```

Función prototipo

Serie de condicionales que permite al cualquier enemigo moverse o caer entre plataformas, marcando el inicio y el final de recorrido. Lleva este nombre por que al principio era una función en prueba que al funcionar gusto tanto que el nombre permaneció.

```
575  int prototipo(int &x, int &y, bool &dw, bool &
576  {
577      int t=0;
578      borrar_enemigo3(x,y);
579      if(y==20||y==30||y==40||y==50){
580              dw=true;
581              dg=true;
582          }
583      if(x>17 && y==10){
584          x--;
585      }
586      if(x<79 && y==20){
587              x++;
588      }
589      if((x==17 && y==10)||(x==17 && y==30)){
590          dw=false;
591      }
592      if((x==79 && y==20)||(x==79 && y==40)){
593          dg=false;
594      }
595      if(x>17 && y==30){
596              x--;
597      }
```

```
622  void enemigos()
623  {
624      if(game_start==true)
625      {
626      prototipo(x_b1,y_b1,dw1,dg1,finA);
627      //--------------
628      if(x_b1==28 && y_b1==20)
629      {
630          inicioB=true;
631      }
632      if(inicioB==true)
633      {
634          prototipo(x_b2,y_b2,dw2,dg2,finB);
635      }
636      //--------------
637      if(x_b2==30 && y_b2==20)
638      {
639          inicioC=true;
640      }
641      if(inicioC==true)
642      {
643          prototipo(x_b3,y_b3,dw3,dg3,finC);
```

Función enemigos

Función en cargada de activar el movimiento de cada barril y reparar la escalera en caso la borre al pasar sobre ella, también es la función encargada de detectar el choque del barril con el jugador.

```
685  int impacto(int &x,int &y,bool &dw,bool &dg)
686  {
687      if(
688          (((y_p1+2)==(y) || (y_p1+2)==(y+1) ||
689              ((x==x_p1) || (x+1 == x_p1+1) ||
690          (((y_p1+3)==(y) || (y_p1+2)==(y+1) ||
691           ((x+1==x_p1) || (x+1 == x_p1+1) || (
692         )
693      {
694          choque_enemigo3(x,y,dw,dg);
695          Explotar();
696          Corazones_p1--;
697          mostrar_salud();
698          Beep(260,200);
699          x=65,y=10;
700      }
701  }
```

Función impacto

Función para validar el choque de los enemigos con el jugador, así como el encargado de restarle un punto de salud.

```
702  void bajar()
703  {
704      if( (x_p1==47 && y_p1==5) )
705      {
706          unico=true;
707      }
708      if(
709          ((x_p1==89 && (y_p1>45 && y_p1<49
710          (x_p1== 7 && (y_p1>34 && y_p1<39)
711          (x_p1==89 && (y_p1>25 && y_p1<29)
712          (x_p1== 7 && (y_p1>14 && y_p1<19)
713          ((x_p1==47 && (y_p1> 4 && y_p1< 9
714          )
715      {
716          eliminar_jugador();
717          y_p1++;
718          pintar_jugador1();
719      }
720      if( (x_p1==47 && y_p1==9) )
721      {
722          unico=false;
723      }
724  }
```

Función bajar

Desciende al jugador de los bloques buff en los que puede pararse sin sufrir daño de caída.

```
727      if(Corazones_p1==1 && cc==1)
728      {
729              xr=(rand()%70)+7;
730          if(xr>10 && xr<19){
731              yr=9;
732          }else if(xr>20 && xr<29){
733              yr=19;
734          }else if(xr>30 && xr<39){
735              yr=29;
736          }else if(xr>40 && xr<49){
737              yr=39;
738          }else if(xr>50 && xr<59){
739              yr=49;
740          }else{
741              yr=19;
742          }
743          gotoxy(xr,yr);printf("+");
744          Sleep(150);
745          gotoxy(xr,yr);printf(" ");
746          cc=0;
747      }
748      if(Corazones_p1==3){
```

Función buff_puerta_escudo

Función encargada de generar la puerta secreta que lleva al jugador, 2 pasos antes de la meta para completar el nivel, esta se genera de forma aleatoria siguiendo el reloj de la computadora como base.

Función Score

Función para asignar puntos al jugador al momento de saltar exitosamente sobre un barril.

```
752  void Score(int x,int y)
753  {
754      gotoxy(85,1);printf("SCORE   %d",puntuacion_p1);
755      if(
756          (((y_p1+4)==(y) || (y_p1+5)==(y+1) || (y_p1+
757           ((x==x_p1)   || (x==x_p1+1))) && (saltod_on
758         )
759         {
760             puntuacion_p1+=100;
761             Beep(500,200);
762         }
763  }
```

```
764  void presentacion()
765  {
766      escenario();
767      gotoxy(85,1);printf("SCORE   %d",puntuacion_p1);
768      gotoxy(37,20);printf("%c%c%c%c%c %c%c %c%c%c%c %
769      gotoxy(37,21);printf("%c%c%c%c%c %c%c %c%c%c%c %
770      gotoxy(37,22);printf("%c%c%c%c%c %c%c %c%c%c%c %
771      gotoxy(37,23);printf("%c%c%c%c%c %c%c %c%c%c%c %
772
773      gotoxy(48,27);puts(jugador1_1);
774      gotoxy(48,28);puts(jugador1_2);
775      gotoxy(48,29);puts(jugador1_3);
776      gotoxy(48,30);puts(jugador1_4);
777
778      Sleep(5000);
779      system("cls");
780  }
```

Función presentación

Se encarga de generar la pantalla de presentación del juego durante unos segundos e imprimir parte del escenario.

Función presentacion2

Se encarga de generar la pantalla de juego completado el cual consiste en agregar al conejo quien fue rescatado.

```
781  void presentacion2()
782  {
783      escenario();
784      gotoxy(85,1);printf("SCORE   %d",puntuacion_p1);
785      gotoxy(37,20);printf("%c%c%c%c%c %c%c %c%c%c%c %
786      gotoxy(37,21);printf("%c%c%c%c%c %c%c %c%c%c%c %
787      gotoxy(37,22);printf("%c%c%c%c%c %c%c %c%c%c%c %
788      gotoxy(37,23);printf("%c%c%c%c%c %c%c %c%c%c%c %
789
790      gotoxy(46,27);puts(jugador1_1);
791      gotoxy(46,28);puts(jugador1_2);
792      gotoxy(46,29);puts(jugador1_3);
793      gotoxy(46,30);puts(jugador1_4);
794
795      x_rabbit=50,y_rabbit=28;pintar_rabbit();
796
797      Sleep(5000);
798      system("cls");
799  }
```

```
800  void interseccion()                                    /
801  {
802      game_start==false;
803      Sleep(1000);
804      for(int i=1;i<10;i++)
805      {
806      if(x_mm>72){borrar_monstruo();x_mm--;pintar_monst
807      }
808      for(int j=1;j<18;j++)
809      {
810      if(x_mm<97){borrar_monstruo();x_mm++;pintar_monst
811      if(x_rabbit<97){borrar_rabbit();x_rabbit++;pintar
812      }
813      borrar_monstruo();borrar_rabbit();
814      system("cls");Nivel++;escenario();
815      presentacion();
816      x_rabbit=67;y_rabbit=6;x_mm=81;y_mm=5;
817      x_p1=9;y_p1=49;
818      x_b1=60;y_b1=10;dg1=true;dw1=true;
819      x_b2=60;y_b2=10;dg2=true;dw2=true;inicioB=false;
820      x_b3=60;y_b3=10;dg3=true;dw3=true;inicioC=false;
821      x_b4=60;y_b4=10;dg4=true;dw4=true;inicioE=false;
```

Función intersección

Esta función se utiliza cuando el personaje llega al punto de meta para generar la animación del monstruo llevándose al conejo, restaura los valores de todos los personajes y barriles

Función felicidades

Función para completar el juego, borra el suelo debajo del monstruo y anima la caída hasta el primer piso, muestra animación de explosión y acaba el juego.

```
827  void felicidades()
828  {
829      game_start==false;
830      Sleep(1000);
831      gotoxy(x_mm-1,y_mm+8);printf("          ");
832      gotoxy(x_mm-1,y_mm+9);printf("          ");
833      Sleep(500);
834      for(int i=1;i<41;i++)
835      {
836      if(y_mm<60){borrar_monstruo();y_mm++;pintar_
837      }
838      borrar_monstruo();
839      explotar_monstruo();
840      system("cls");
841      presentacion2();
842  }
```

```
843  void mennu()
844  {
845      gotoxy(37,20);printf("%c%c%c%c%c %c%c %c%c
846      gotoxy(37,21);printf("%c%c%c%c%c %c%c %c%c
847      gotoxy(37,22);printf("%c%c%c%c%c %c%c %c%c
848      gotoxy(37,23);printf("%c%c%c%c%c %c%c %c%c
849
850      bool repite=true;
851      int opcion;
852
853      const char *titulo="MENU DE INICIO";
854      const char *opciones[]={"Nuevo Juego","Con
855      int n = 4;
856
857      do {
858          limpiar(41,25);
859          opcion = eleccion(titulo,opciones,n);
860          switch (opcion) {
861              case 1:
862                  Nueva_partida();
863                  break;
```

Función mennu

Función para mostrar el menú de inicio del juego para crear una nueva partida, cargar los datos de la partida anterior con nombre de usuario o Salir del juego.

```
884  void Nueva_partida()
885  {
886      bool repite=true;
887      int opcion;
888
889      ifstream fentrada("caset.bin",ios::in | ios::bin
890      fentrada.read(reinterpret_cast<char *>(&partidas
891      fentrada.read(reinterpret_cast<char *>(&partidas
892
893      const char *titulo="Nueva Partida....?   ";
894      const char *opciones[]={partidas[0].nombre,parti
895      int n = 3;
896
897      fentrada.close();
898      do {
899          opcion = eleccion(titulo,opciones,n);
900          switch (opcion) {
901              case 1:
902                  slot=0;
903                  Newprofile();
904                  break;
905              case 2:
906                  slot=1;
```

Función Nueva_partida

Función para crear una nueva partida en la que se ingrese un nuevo nombre de usuario con los datos iniciales. Esta función muestra los espacios donde puede guardase la partida en un fichero binario.

```
919  void Newprofile()
920  {
921      gotoxy(41,25);cout<<"Cual es tu nombre?";
922
923      gotoxy(41,28);cout<<"> ";
924      if(slot==0)
925      {
926          ofstream fsalida("caset.bin",ios::out | ic
927          cin>>partidas[0].nombre;
928          partidas[0].puntaje=0;
929          partidas[0].vidas=3;
930          partidas[0].salud=3;
931
932          fsalida.write(reinterpret_cast<char *>(&pe
933          fsalida.write(reinterpret_cast<char *>(&pe
934          fsalida.close();
935      }
936      if(slot==1)
937      {
938          ofstream fsalida("caset.bin",ios::out | ic
939          cin>>partidas[1].nombre;
940          partidas[1].puntaje=0;
```

Función Newprofile

Función para ingresar los datos de una nueva partida de acuerdo a la ubicación seleccionada en el menú anterior.

Función Continuar

Función para continuar una partida con los datos almacenados en la ubicación seleccionada en ese mismo menú.

```
951  void Continuar()
952  {
953      bool repite=true;
954      int opcion;
955
956      const char *titulo="Continuar....?   ";
957      const char *opciones[]={partidas[0].nombre,p
958      int n = 3;
959
960      do {
961          opcion = eleccion(titulo,opciones,n);
962          switch (opcion) {
963              case 1:
964                  slot=0;
965
966                  repite=false;
967                  break;
968              case 2:
969                  slot=1;
970
971                  repite=false;
972                  break;
973              case 3:
```

```
979  void Puntuaciones()
980  {
981      bool repite=true;
982      int opcion;
983      ifstream fentrada("caset.bin",ios::in | ios:
984      fentrada.read(reinterpret_cast<char *>(&part
985      fentrada.read(reinterpret_cast<char *>(&part
986
987      const char *titulo="Continuar....?    ";
988      const char *opciones[]={partidas[0].nombre,p
989      int n = 3;
990      fentrada.close();
991      do {
992          opcion = eleccion(titulo,opciones,n);
993          gotoxy(60,27);cout<<partidas[0].puntaje;
994          gotoxy(60,28);cout<<partidas[1].puntaje;
995          switch (opcion) {
996              case 3:
997                  repite = false;
998                  break;
999        }
000    } while(repite);
001  }
```

Función Puntuaciones

Función para mostrar las máximas puntuaciones en el menú principal.

Función eleccion

Función que determina la opción utilizada en el menú mediante una flecha la cual según donde se detenga y al presionar la tecla Enter, el usuario ingresara a la opción donde la flecha se encuentre.

```
1002  int eleccion(const char titulo[], const char *op
1003  {
1004      int opcionSeleccionada = 1;
1005      int tecla;
1006      bool repite = true;
1007
1008      gotoxy(41,27);printf("                    ");
1009      gotoxy(41,28);printf("                    ");
1010      gotoxy(41,29);printf("                    ");
1011      gotoxy(41,30);printf("                    ");
1012
1013      do{
1014          gotoxy(37, 26 + opcionSeleccionada); cou
1015          gotoxy(41,25); cout << titulo;
1016
1017          for (int i = 0; i < n; ++i) {
1018              gotoxy(41,27 + i); cout << i + 1 <<
1019          }
1020          do {
1021              tecla = getch();
1022          } while (tecla != ARRIBA && tecla != ABA
1023
1024          switch (tecla) {
```

```
1048  void guardardatos()
1049  {
1050      if(nuevaentrada==true)
1051      {
1052          puntuacion_p1=0;
1053          Num_vidas_p1=3;
1054          Corazones_p1=3;
1055      }
1056      if(slot==0)
1057      {
1058          ofstream fsalida("caset.bin",ios::out | ios
1059              puntuacion_p1=partidas[0].puntaje;
1060              Corazones_p1=partidas[0].salud;
1061              fsalida.write(reinterpret_cast<char *>(
1062              fsalida.write(reinterpret_cast<char *>(
1063              fsalida.close();
1064      }
1065      if(slot==1)
1066      {
1067              ofstream fsalida("caset.bin",ios::out |
1068              puntuacion_p1=partidas[1].puntaje;
1069              Corazones_p1=partidas[1].salud;
1070              fsalida.write(reinterpret_cast<char *>(
```

Función guardatos

Función para asignar datos del fichero binario a la consola de ejecución

Función asignardatos

Función para almacenar los datos de las partidas en el archivo binario,

```
1075  void asignardatos()
1076  {
1077          if(slot==0)
1078          {
1079              ofstream fsalida("caset.bin",ios::out |
1080              partidas[0].puntaje=puntuacion_p1;
1081              partidas[0].salud=Corazones_p1;
1082              fsalida.write(reinterpret_cast<char *>(&
1083              fsalida.write(reinterpret_cast<char *>(&
1084              fsalida.close();
1085          }
1086          if(slot==1)
1087          {
1088              ofstream fsalida("caset.bin",ios::out |
1089              partidas[1].puntaje=puntuacion_p1;
1090              partidas[1].salud=Corazones_p1;
1091              fsalida.write(reinterpret_cast<char *>(&
1092              fsalida.write(reinterpret_cast<char *>(&
1093              fsalida.close();
1094          }
1095
1096  }
```

```
1097  void limpiar(int x,int y)
1098  {
1099      gotoxy(x,y);  cout<<"                    ";
1100      gotoxy(x,y+2);cout<<"                    ";
1101      gotoxy(x,y+3);cout<<"                    ";
1102      gotoxy(x,y+4);cout<<"                    ";
1103      gotoxy(x,y+5);cout<<"                    ";
1104      gotoxy(x,y+6);cout<<"                    ";
1105  }
```

Función limpiar

Función empleada en el menú de inicio para navegar por los menús sin dejar rastro de los resultados.

Función jugar

Función paralela que se encarga del movimiento del jugador junto a algunas condicionales para que el juego termine.

```
1106  void jugar(void)
1107  {
1108      game_start=true;
1109      if(kbhit())
1110      {
1111          unsigned char tecla = getch();
1112          switch(tecla)
1113          {
1114          case IZQUIERDA:
1115              if(
1116                  ((x_p1>=8 && x_p1<=93) && (y_p1==
1117                  ((x_p1>=8 && x_p1<=93) && (y_p1==
1118                  ((x_p1>=4 && x_p1<=93) && (y_p1==
1119                  ((x_p1>=8 && x_p1<=93) && (y_p1==
1120                  ((x_p1>=4 && x_p1<=93) && (y_p1==
1121                  ((x_p1>=42 && x_p1<=63) && (y_p1=
1122                 )
1123              {
1124                  eliminar_jugador();
1125                  x_p1-=2;   //ix=ix-2 es lo mismo
1126                  pintar_jugador1();
1127              }
1128              break;
```

Función main

Función principal encargada de cargar nuestras funciones, variables y métodos durante la ejecución del programa. En ella se encuentra un bucle while, el cual ejecuta el juego hasta que el personaje pierda todas sus vidas o complete el juego.

```
1224  int main(){
1225      system("color 79");
1226      ocultarcursor();
1227      srand(time(NULL));
1228      escenario();
1229      mennu();
1230      if(salir_del_juego==false)
1231      {
1232      game_over=false;
1233      guardardatos();
1234      presentacion();
1235      pintar_plataforma();
1236      mostrar_escalera();
1237      pintar_jugador1();
1238
1239      while(game_over!=true){
1240          bajar();
1241          buff_puerta_escudo();
1242          muerte_caida();
1243          mover_monstruo();
1244          jugar();
1245          enemigos();
1246          mostrar_escalera();
```