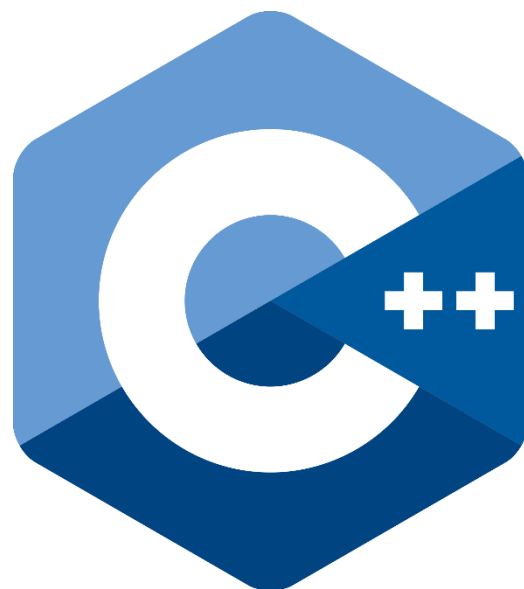


4 de enero de 2023

UF2 – OPTIMIZACIÓN DE SOFTWARE

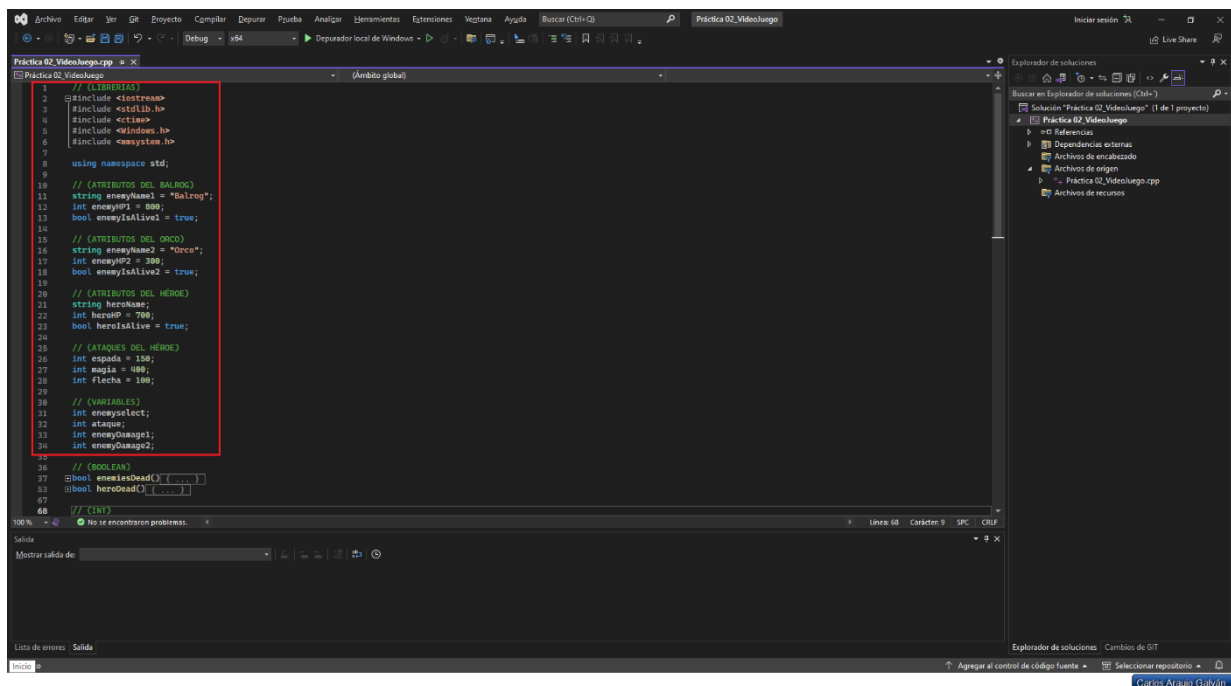
PRÁCTICA 04 – REFACTORIZACIÓN DE CÓDIGO



Carlos Araujo Galván
1º DAW / CURSO 2022-2023

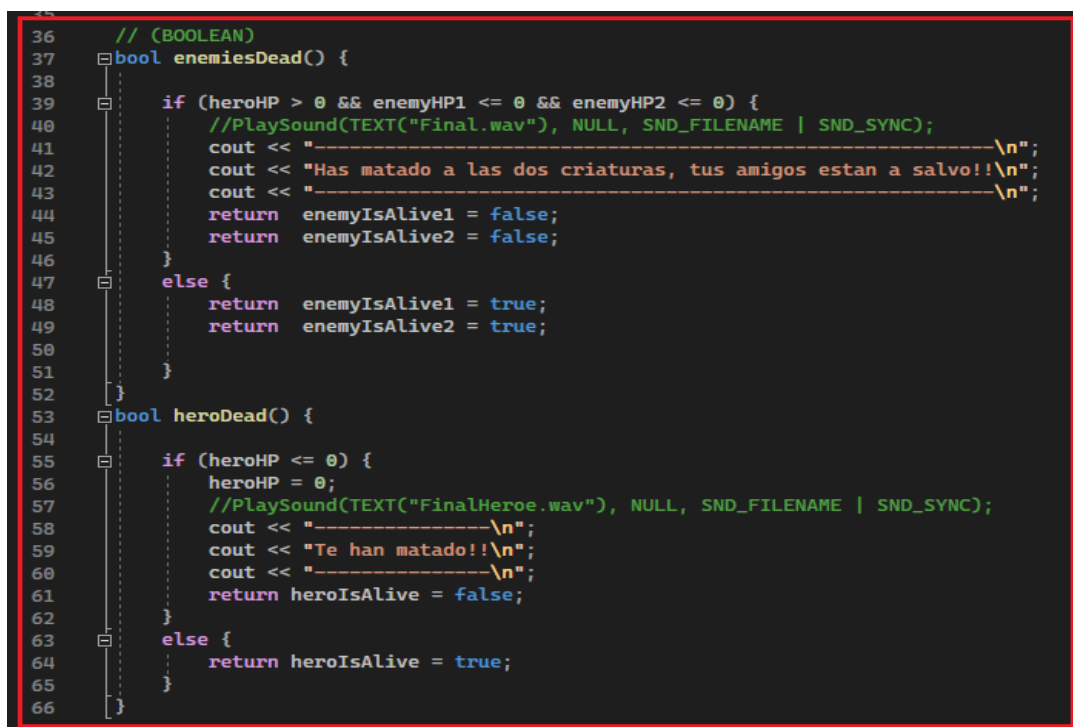


- 1) Antes de comenzar explicando y analizando el código de mi videojuego, voy a mostrar las correspondientes **librerías** que he aplicado, los **atributos del enemigo 1**, los **atributos del enemigo 2**, los **atributos del héroe**, los correspondientes **ataques del héroe** y las **variables** que se han utilizado en el videojuego:



```
1 // LIBRERÍAS
2 #include <iostream>
3 #include <stdlib.h>
4 #include <ctime>
5 #include <unistd.h>
6 #include <csystem.h>
7
8 using namespace std;
9
10 // ATRIBUTOS DEL ENEMIGO
11 string enemyName1 = "Balrog";
12 int enemyHP1 = 800;
13 bool enemyIsAlive1 = true;
14
15 // ATRIBUTOS DEL ENEMIGO 2
16 string enemyName2 = "Orco";
17 int enemyHP2 = 300;
18 bool enemyIsAlive2 = true;
19
20 // ATRIBUTOS DEL HÉROE
21 string heroName;
22 int heroHP = 700;
23 bool heroIsAlive = true;
24
25 // ATAQUES DEL HÉROE
26 int espada = 150;
27 int magia = 400;
28 int flecha = 100;
29
30 // VARIABLES
31 int enemySelect;
32 int enemy;
33 int enemyDamage1;
34 int enemyDamage2;
35
36 // (BOOLEAN)
37 bool enemiesDead() {
38
39 }
40
41 // (BOOLEAN)
42 bool heroDead() {
43
44 }
```

- 2) Por orden siempre, en primer lugar, he añadido los **booleanos**, para indicar en todo momento el **estatus** de los personajes, si nos devuelve un **false**, el personaje **morirá**, si nos devuelve un **true** el personaje **continuará** la pelea, estos dos booleanos están añadidos en su correspondiente función:
- NOTA:** Estos booleanos son añadidos solamente cuando el videojuego ha finalizado.



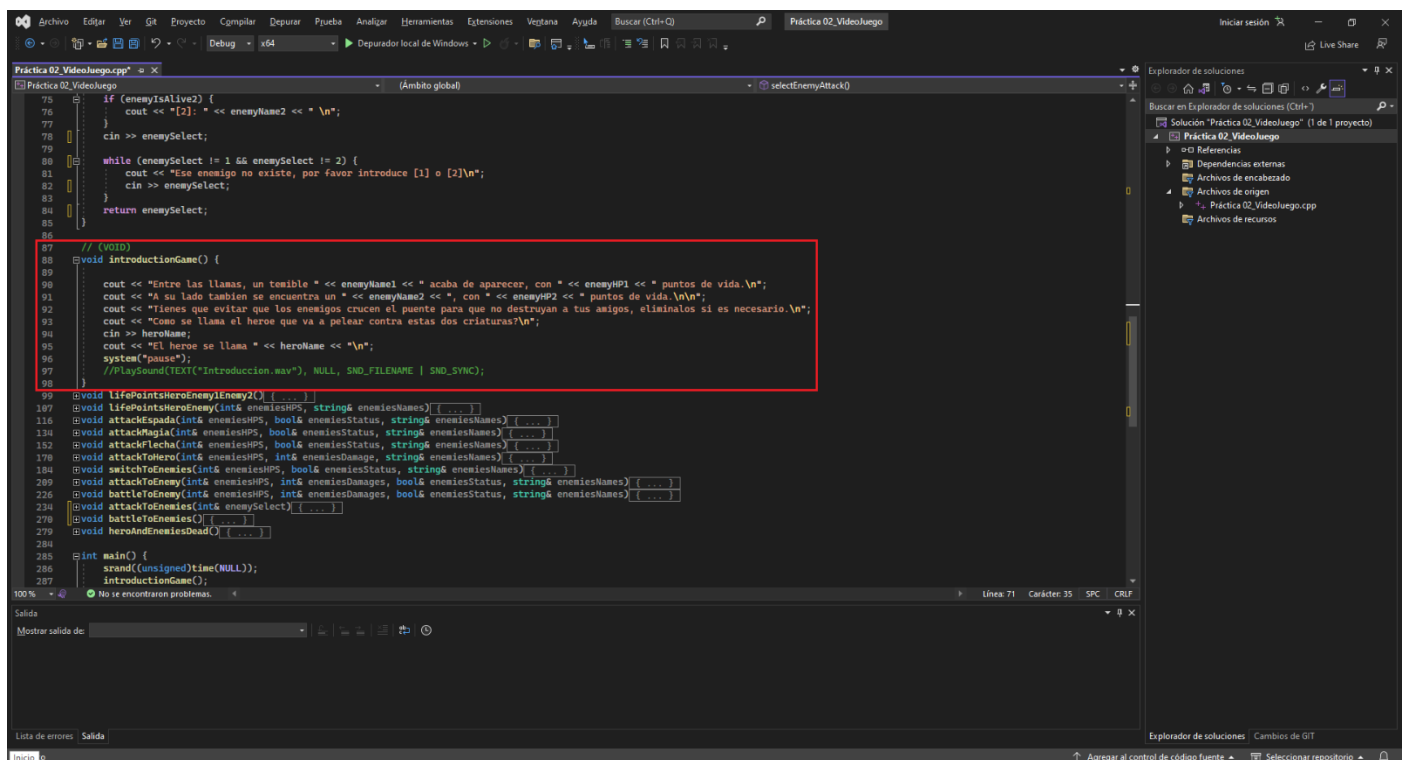
```
36 // (BOOLEAN)
37 bool enemiesDead() {
38
39     if (heroHP > 0 && enemyHP1 <= 0 && enemyHP2 <= 0) {
40         //PlaySound(TEXT("Final.wav"), NULL, SND_FILENAME | SND_SYNC);
41         cout << "-----\n";
42         cout << "Has matado a las dos criaturas, tus amigos estan a salvo!!\n";
43         cout << "-----\n";
44         return enemyIsAlive1 = false;
45         return enemyIsAlive2 = false;
46     }
47     else {
48         return enemyIsAlive1 = true;
49         return enemyIsAlive2 = true;
50     }
51 }
52
53 bool heroDead() {
54
55     if (heroHP <= 0) {
56         heroHP = 0;
57         //PlaySound(TEXT("FinalHeroe.wav"), NULL, SND_FILENAME | SND_SYNC);
58         cout << "-----\n";
59         cout << "Te han matado!!\n";
60         cout << "-----\n";
61         return heroIsAlive = false;
62     }
63     else {
64         return heroIsAlive = true;
65     }
66 }
```



- 3) En este apartado, he añadido como variable **INT**, la selección del **enemySelect**, para que nos **devuelva** la **elección del enemigo a atacar**, este **INT** la he añadido a su **función** correspondiente:

```
68 // (INT)
69 int selectEnemyAttack() {
70     cout << "A que enemigo quieres atacar?\n";
71     if (enemyIsAlive1) {
72         cout << "[1]: " << enemyName1 << " \n";
73     }
74     if (enemyIsAlive2) {
75         cout << "[2]: " << enemyName2 << " \n";
76     }
77     cin >> enemysselect;
78     while (enemysselect != 1 && enemysselect != 2) {
79         cout << "Ese enemigo no existe, por favor introduce [1] o [2]\n";
80         cin >> enemysselect;
81     }
82     return enemysselect;
83 }
84
85
```

- 4) Aquí tenemos todos los **VOID**, con sus **funciones** y sus **parámetros** correspondientes, los iré analizando y explicando uno a uno. En primer lugar, tenemos la **introducción del videojuego**, donde se nos pedirá como se va a **llamar** nuestro **héroe**, esta introducción la tengo asignada como **VOID**, ya que no nos devuelve **ningún valor**, está añadida en su función correspondiente nombrada como **introductionGame**:



- 5) También he añadido al videojuego, que, por cada turno del combate, nos muestre siempre los **puntos de vida de los personajes**, en la **primera función** se puede apreciar donde nos mostrará los puntos de vida de los **tres personajes**, siempre y cuando estén los **tres vivos**, en el momento en que **derrotamos a un enemigo**, pasará a mostrarnos la **segunda función**, junto a sus **parámetros** correspondientes, para así mostrarnos automáticamente los puntos de vida **solamente** del enemigo que **esté vivo**, y ahorrarnos código:

```
99 void lifePointsHeroEnemy1Enemy2() {
100
101     cout << "-----\n";
102     cout << enemyName1 << " " << enemyHP1 << " Puntos de vida\n";
103     cout << enemyName2 << " " << enemyHP2 << " Puntos de vida\n";
104     cout << heroName << " " << heroHP << " Puntos de vida\n";
105     cout << "-----\n";
106 }
107 void lifePointsHeroEnemy(int& enemiesHPS, string& enemiesNames) {
108
109     cout << "-----\n";
110     cout << enemiesNames << " " << enemiesHPS << " Puntos de vida\n";
111     cout << heroName << " " << heroHP << " Puntos de vida\n";
112     cout << "-----\n";
113
114     cout << "Ya solamente te queda por matar al " << enemiesNames << "\n";
115 }
```

- 6) En este apartado, he añadido los **tres ataques del héroe** hacia los enemigos, los he agrupado en la **función del ataque** que corresponde, junto a sus **parámetros** para así **agrupar** a los dos enemigos en una **única función**.

NOTA: En cada ataque, he añadido también el **booleano** que nos va a indicar si hemos **derrotado** en ese instante a ese **enemigo o no**:

```
116 void attackEspada(int& enemiesHPS, bool& enemiesStatus, string& enemiesNames) {
117
118     enemiesHPS = enemiesHPS - espada;
119     if (enemiesHPS <= 0) {
120         enemiesHPS = 0;
121     }
122     cout << "Has atacado con la espada\n";
123     cout << "-----\n";
124     cout << "Le has hecho " << espada << " puntos de danyo\n";
125     cout << "Al " << enemiesNames << " le quedan " << enemiesHPS << " puntos de vida\n";
126     cout << "-----\n";
127     if (enemiesHPS <= 0) {
128         enemiesHPS = 0;
129         enemiesStatus = false;
130         cout << "Has matado al " << enemiesNames << "\n";
131         cout << "-----\n";
132     }
133 }
134 void attackMagia(int& enemiesHPS, bool& enemiesStatus, string& enemiesNames) { ... }
152 void attackFlecha(int& enemiesHPS, bool& enemiesStatus, string& enemiesNames) {
153
154     enemiesHPS = enemiesHPS - flecha;
155     if (enemiesHPS <= 0) {
156         enemiesHPS = 0;
157     }
158     cout << "Has atacado con la flecha\n";
159     cout << "-----\n";
160     cout << "Le has hecho " << flecha << " puntos de danyo\n";
161     cout << "Al " << enemiesNames << " le quedan " << enemiesHPS << " puntos de vida\n";
162     cout << "-----\n";
163     if (enemiesHPS <= 0) {
164         enemiesHPS = 0;
165         enemiesStatus = false;
166         cout << "Has matado al " << enemiesNames << "\n";
167         cout << "-----\n";
168     }
169 }
```



- 7) Ahora vamos con el **ataque de los enemigos**, en este caso es un ataque **random**, nos van a hacer un **daño aleatorio**, he añadido **parámetros** para así **agrupar el ataque de los dos enemigos**, en una **única función** junto al **booleano** que nos va a indicar si el héroe ha sido **derrotado en ese ataque o no**:

```
170 void attackToHero(int& enemiesHPS, int& enemiesDamage, string& enemiesNames) {
171
172     if (enemiesHPS > 0) {
173         enemiesDamage = rand() % 200 + 1;
174         heroHP = heroHP - enemiesDamage;
175         if (heroHP <= 0) {
176             heroHP = 0;
177             heroIsAlive = false;
178         }
179         cout << "El " << enemiesNames << " te ha hecho un ataque de " << enemiesDamage << " puntos de danyo\n";
180         cout << "Te quedan " << heroHP << " puntos de vida\n";
181         cout << "-----\n";
182     }
183 }
```

- 8) En los **ataques del héroe**, los he añadido en un **switch** para **indicar** que **ataque** queremos **utilizar** contra el enemigo, he añadido **parámetros**, para agrupar el **ataque hacia los dos enemigos**, en una **única función**, así en este caso da igual a que enemigo vamos a atacar, que va a funcionar correctamente, **independientemente del enemigo** que sea:

```
184 void switchToEnemies(int& enemiesHPS, bool& enemiesStatus, string& enemiesNames) {
185
186     switch (ataque) {
187
188     case 1:
189         //PlaySound(TEXT("Espada.wav"), NULL, SND_FILENAME | SND_SYNC);
190         attackEspada(enemiesHPS, enemiesStatus, enemiesNames);
191         break;
192
193     case 2:
194         //PlaySound(TEXT("Magia.wav"), NULL, SND_FILENAME | SND_SYNC);
195         attackMagia(enemiesHPS, enemiesStatus, enemiesNames);
196         break;
197
198     case 3:
199         //PlaySound(TEXT("Flecha.wav"), NULL, SND_FILENAME | SND_SYNC);
200         attackFlecha(enemiesHPS, enemiesStatus, enemiesNames);
201         break;
202
203     default:
204         cout << "No dispones de esa habilidad, has perdido el turno\n";
205         cout << "-----\n";
206         break;
207     }
208 }
```



- 9) Ahora vamos con la **selección del enemigo**, en este caso he añadido **dos IF** para cada enemigo, dentro de cada **IF**, he añadido las funciones del **switch** con los **tres ataques del héroe**, y **dos funciones** para cada **ataque del enemigo** que vaya a atacar al héroe. Todo esto está agrupado en una **única función** junto a su parámetro **INT&** para la selección del enemigo:

```
234 void attackToEnemies(int& enemySelect) {
235
236     if (enemySelect == 1) {
237         cout << "-----\n";
238         cout << "Que ataque quieres utilizar contra el Balrog?\n";
239         cout << "-----\n";
240         cout << "[1] Golpe con espada\n";
241         cout << "[2] Hechizo de magia\n";
242         cout << "[3] Lanzar una flecha\n";
243         cout << "-----\n";
244
245         cin >> ataque;
246
247         switchToEnemies(enemyHP1, enemyIsAlive1, enemyName1);
248         attackToHero(enemyHP1, enemyDamage1, enemyName1);
249         attackToHero(enemyHP2, enemyDamage2, enemyName2);
250     }
251
252     if (enemySelect == 2) {
253         cout << "-----\n";
254         cout << "Que ataque quieres utilizar contra el Orco?\n";
255         cout << "-----\n";
256         cout << "[1] Golpe con espada\n";
257         cout << "[2] Hechizo de magia\n";
258         cout << "[3] Lanzar una flecha\n";
259         cout << "-----\n";
260
261         cin >> ataque;
262
263         switchToEnemies(enemyHP2, enemyIsAlive2, enemyName2);
264         attackToHero(enemyHP1, enemyDamage1, enemyName1);
265         attackToHero(enemyHP2, enemyDamage2, enemyName2);
266     }
267 }
268
269 }
```

- 10) En esta función tenemos lo mismo, pero esta vez con la **batalla del héroe y un único enemigo**, he añadido los **parámetros** correspondientes para **agrupar** a los **dos enemigos** en una **única función**, así dará igual que enemigo es el que se **mantiene vivo**, que el videojuego nos lo mostrará correctamente:

```
209 void attackToEnemy(int& enemiesHPS, int& enemiesDamages, bool& enemiesStatus, string& enemiesNames) {
210
211     if (enemiesStatus) {
212
213         cout << "Que ataque quieres utilizar?\n";
214         cout << "-----\n";
215         cout << "[1] Golpe con espada\n";
216         cout << "[2] Hechizo de magia\n";
217         cout << "[3] Lanzar una flecha\n";
218         cout << "-----\n";
219
220         cin >> ataque;
221
222         switchToEnemies(enemiesHPS, enemiesStatus, enemiesNames);
223         attackToHero(enemiesHPS, enemiesDamages, enemiesNames);
224     }
225 }
```



11) En esta función, estoy indicando la **batalla contra un solo enemigo**, el que se haya **mantenido vivo**, en este caso he añadido **parámetros**, para agrupar a las **funciones** que ya tenían sus **parámetros añadidos anteriormente**, estas **funciones** serían los puntos de vida del héroe y del único enemigo vivo, y de los ataques que se emplean hacia el enemigo vivo y del enemigo hacia el héroe, así he podido **agrupar todo el combate** del héroe contra el enemigo vivo, en **una única función**:

```
226 void battleToEnemy(int& enemiesHPS, int& enemiesDamages, bool& enemiesStatus, string& enemiesNames) {
227
228     lifePointsHeroEnemy(enemiesHPS, enemiesNames);
229     attackToEnemy(enemiesHPS, enemiesDamages, enemiesStatus, enemiesNames);
230
231     system("pause");
232     system("cls");
233 }
```

12) Aquí tenemos lo mismo que antes, pero esta vez tenemos la **batalla** de los **tres personajes vivos**, tenemos la **función** con sus respectivos **puntos de vida**, la **función** de la **selección del enemigo** a atacar, de los **ataques del héroe**, junto al **switch** y de los **ataques de los dos enemigos** hacia el héroe.

Todo esto agrupado a una **única función**:

```
270 void battleToEnemies() {
271
272     lifePointsHeroEnemy1Enemy2();
273     selectEnemyAttack();
274     attackToEnemies(enemySelect);
275
276     system("pause");
277     system("cls");
278 }
```

13) Por último, he **agrupado las dos funciones** en el que dentro se encontraban los **booleanos**, donde nos van a indicar la **finalización** del videojuego junto a la **muerte del héroe**, o del **último enemigo muerto**

Todo esto agrupado a una **única función**:

```
279 void heroAndEnemiesDead() {
280
281     enemiesDead();
282     heroDead();
283 }
```



14) Y este sería el **resultado final** de mi código **Main** del videojuego:

```
30 // (VARIABLES)
31 int enemySelect;
32 int ataque;
33 int enemyDamage1;
34 int enemyDamage2;
35
36 // (BOOLEAN)
37 bool enemiesDead() { ... }
53 bool heroDead() { ... }
67
68 // (INT)
69 int selectEnemyAttack() { ... }
86
87 // (VOID)
88 void introductionGame() { ... }
99 void lifePointsHeroEnemy1Enemy2() { ... }
107 void lifePointsHeroEnemy(int& enemiesHPS, string& enemiesNames) { ... }
116 void attackEspada(int& enemiesHPS, bool& enemiesStatus, string& enemiesNames) { ... }
134 void attackMagia(int& enemiesHPS, bool& enemiesStatus, string& enemiesNames) { ... }
152 void attackFlecha(int& enemiesHPS, bool& enemiesStatus, string& enemiesNames) { ... }
170 void attackToHero(int& enemiesHPS, int& enemiesDamage, string& enemiesNames) { ... }
184 void switchToEnemies(int& enemiesHPS, bool& enemiesStatus, string& enemiesNames) { ... }
209 void attackToEnemies(int& enemySelect) { ... }
245 void attackToEnemy(int& enemiesHPS, int& enemiesDamages, bool& enemiesStatus, string& enemiesNames) { ... }
262 void battleToEnemies() { ... }
271 void battleToEnemy(int& enemiesHPS, int& enemiesDamages, bool& enemiesStatus, string& enemiesNames) { ... }
279 void heroAndEnemiesDead() { ... }
284
285 int main() {
286     srand((unsigned)time(NULL));
287     introductionGame();
288
289     while (heroIsAlive && enemyIsAlive1 && enemyIsAlive2) {
290         battleToEnemies();
291     }
292     while (heroIsAlive && enemyIsAlive1) {
293         battleToEnemy(enemyHP1, enemyDamage1, enemyIsAlive1, enemyName1);
294     }
295     while (heroIsAlive && enemyIsAlive2) {
296         battleToEnemy(enemyHP2, enemyDamage2, enemyIsAlive2, enemyName2);
297     }
298     heroAndEnemiesDead();
299 }
```

Carlos Araujo Galván

