
	Modulo: 5		
	UF:2 Practica 4 Refactorización de código		
Apellidos: <i>Martínez Méndez</i>	Nombre: <i>Mario Alejandro</i>	Fecha:15/01/2023	

Objetivo de la practica

El objetivo principal de esta práctica es aprender a refactorizar nuestro código, aplicando las nuevas técnicas y habilidades adquiridas, como los parámetros por referencia.

Refactorización del Código del Videojuego

El video juego estaba compuesto por una serie de funciones que se repetían para cada enemigo, es decir, funciones como la de estatus del enemigo o funciones de ataque, tenían una copia de sí misma pero con diferentes variables.

En la refactorización del videojuego, eliminé casi todas las funciones o procesos que se repetían, utilizando los parámetros por referencia (&).

```

+void gameStart() { ... }
+void wichEnemy() { ... }
+bool enemyStatus1() { ... }
+int battleAttack1() { ... }
+void battle1() { ... }

+void enemyAttacksBack1() { ... }
+bool enemyStatus2() { ... }
+int battleAttack2() { ... }
+void battle2() { ... }
+void enemyAttacksBack2() { ... }

```

Ilustración 1 Funciones antes de refactorizar (10)

```

+void gameStart() { ... }
+void wichEnemy() { ... }
+void enemyStatus(string enemyName, int& enemyHP, bool& enemyisAlive) { ... }
+int battleAttack() { ... }
+void battle(int& enemyHP) { ... }
+void enemyAttacksBack(string enemyName, int& enemyAttack) { ... }
+void startbattle(string name, bool& enemyisAlive, int& enemyAttack) { ... }

```

Ilustración 2 Funciones después de refactorizar (7)



Funciones eliminadas

```

bool enemyStatus1() {
    if (enemyHP <= 0)
    {
        cout << enemyName1 << " Ha muerto \n";
        return false;
    }
    else {
        cout << "Al enemigo " << enemyName1 << " le quedan " << enemyHP << " Puntos de vida \n";
        return true;
    }
}

```

Ilustración 3 Función para revisar el estatus de la vida

	Modulo: 5		
	UF:2 Practica 4 Refactorización de código		
Apellidos: <i>Martínez Méndez</i>	Nombre: <i>Mario Alejandro</i>	Fecha:15/01/2023	

```

void battle1() {
    if (heroAttack == 1) {
        heroAttack = espadaAttack;
        espadaAttack = 10 + rand() % 36;
        enemyHP = enemyHP - espadaAttack;
        /*bool opening = PlaySound(TEXT("laser1.wav"), NULL, SND_SYNC);*/
    }
    if (heroAttack == 2) {
        heroAttack = magiaAttack;
        magiaAttack = 20 + rand() % 121;
        enemyHP = enemyHP - magiaAttack;
        contadorMagia = contadorMagia - 1;
        /*bool opening = PlaySound(TEXT("rocket3.wav"), NULL, SND_SYNC);*/
        if (contadorMagia == 0) {
            cout << "Ya no tienes ataques de lanza granadas\n";
        }
    }
    battleAttack1();

    enemyisAlive = enemyStatus1();
}

```

Ilustración 4 Función de ataque al enemigo

```

void enemyAttacksBack1() {
    if (heroHP <= 0) {
        heroHP = 0;
        cout << "El enemigo " << enemyName1 << " te ha matado \n";
        cout << "Game Over \n";
        heroisAlive = false;
    }
    else {
        cout << "El enemigo " << enemyName1 << " te ha hecho un ataque de " << enemyDamage << " puntos de danyo, te quedan " << heroHP << " puntos de vida \n";
    }
    if (heroisAlive && heroHP <= 100 && contadorLife > 0) {
        cout << "Tu vida esta baja. Deseas curarte? \n";
        cout << "[Y] si \n";
        cout << "[N] no \n";

        cin >> cura;
        cura = (char)toupper('y');
        if (contadorLife == 0) {
            cout << "Ya no tienes mas paquetes medicos\n";
        }
        contadorLife = contadorLife - 1;
        if (cura == "Y") {
            heroHP = heroHP + paqueteMedico;
            cout << "Recuperaste " << paqueteMedico << " de puntos de vida, ahora te quedan " << heroHP << " de puntos de vida \n";
        }
    }
}

```

Ilustración 5 Función que resta vida al héroe



```

int battleAttack1() {

    enemyDamage = 1 + rand() % 71;
    enemyDamage2 = 50 + rand() % 111;
    return enemyDamage, enemyDamage2;
}

```

Ilustración 6 Función de ataques del enemigo

	Modulo: 5		
	UF:2 Practica 4 Refactorización de código		
Apellidos: <i>Martínez Méndez</i>	Nombre: <i>Mario Alejandro</i>	Fecha:15/01/2023	

La segunda parte de la refactorización se centro en reducir el contenido del main, creando una función para llamar a la fase de batalla del juego, justo después de seleccionar al enemigo que se desea atacar y el ataque que se desea usar.

```
while (enemyisAlive == true || enemyisAlive2 == true && heroisAlive == true) {

    if (enemyisAlive == true || enemyisAlive2 == true && heroisAlive == true) {
        wichEnemy();

        if (enemyName == "1") {
            if (enemyisAlive == true) {
                battle1();
                /*enemyStatus1();*/

                if (enemyisAlive) {
                    //Ataca al héroe
                    heroHP = heroHP - enemyDamage;
                    enemyAttacksBack1();
                }
            }
        }

        if (enemyName == "2") {
            if (enemyisAlive2 == true) {
                battle2();
                /*enemyStatus2();*/

                if (enemyisAlive2) {
                    //Ataca al héroe
                    heroHP = heroHP - enemyDamage2;
                    enemyAttacksBack2();
                }
            }
        }
    }
}
```

Ilustración 7 Main antes de refactorizar

```
int main()
{
    gameStart();
    srand(time(NULL));
    while (enemyisAlive1 == true || enemyisAlive2 == true && heroisAlive == true) {
        if (enemyisAlive1 == true || enemyisAlive2 == true && heroisAlive == true) {
            wichEnemy();
            startbattle(name, enemyisAlive1, enemyAttack1);
        }

        if (heroHP <= 0) {
            break;
        }
    }
}
```

Ilustración 8 Main refactorizado

Después de la refactorización se puede observar una disminución considerable de líneas con respecto a la segunda versión del juego. La segunda versión tiene, 258 líneas y la tercera 187 líneas.