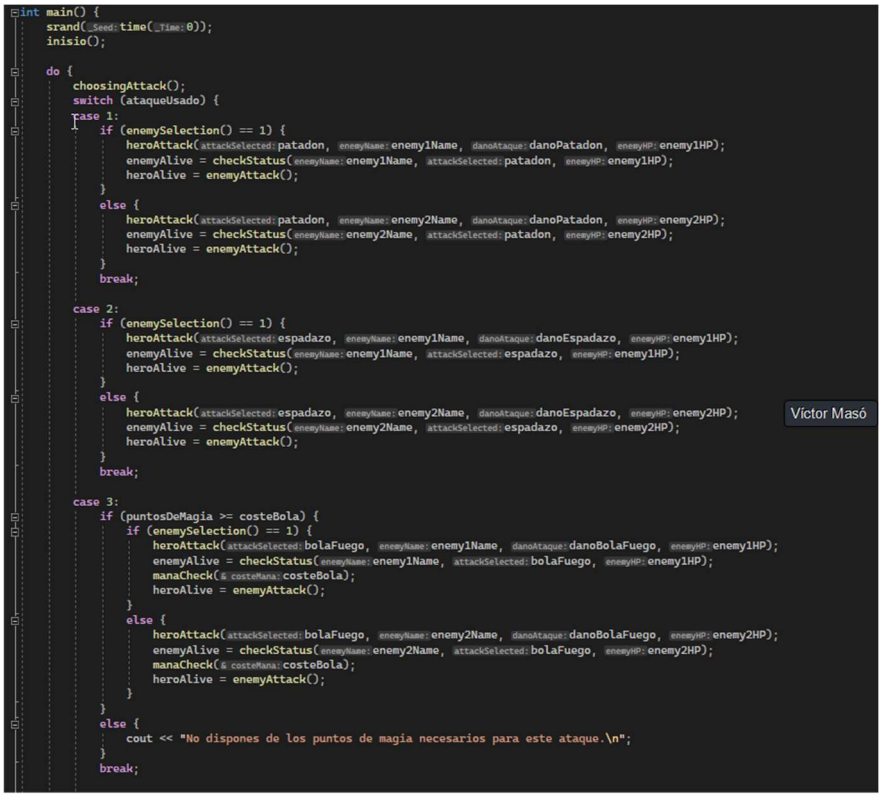
M5-UF2-Pr04

Juego M5 - Update 1.4

En esta actualización he optado por crear un código bastante mas limpio gracias al uso de funciones y a pasar los parámetros por referencia dentro de estas funciones.



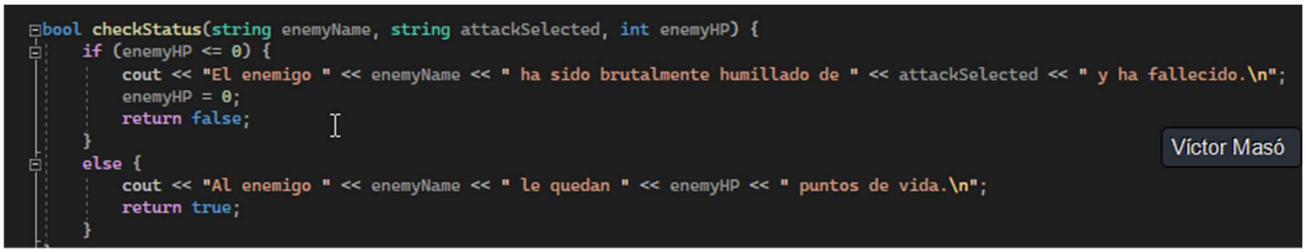

```
int main() {
    srand(_Seed: time(_Time: 0));
    inisio();

    do {
        choosingAttack();
        switch (ataqueUsado) {
        case 1:
            if (enemySelection() == 1) {
                heroAttack(attackSelected: patadon, enemyName: enemy1Name, danoAtaque: danoPatadon, enemyHP: enemy1HP);
                enemyAlive = checkStatus(enemyName: enemy1Name, attackSelected: patadon, enemyHP: enemy1HP);
                heroAlive = enemyAttack();
            }
            else {
                heroAttack(attackSelected: patadon, enemyName: enemy2Name, danoAtaque: danoPatadon, enemyHP: enemy2HP);
                enemyAlive = checkStatus(enemyName: enemy2Name, attackSelected: patadon, enemyHP: enemy2HP);
                heroAlive = enemyAttack();
            }
            break;

        case 2:
            if (enemySelection() == 1) {
                heroAttack(attackSelected: espadazo, enemyName: enemy1Name, danoAtaque: danoEspadazo, enemyHP: enemy1HP);
                enemyAlive = checkStatus(enemyName: enemy1Name, attackSelected: espadazo, enemyHP: enemy1HP);
                heroAlive = enemyAttack();
            }
            else {
                heroAttack(attackSelected: espadazo, enemyName: enemy2Name, danoAtaque: danoEspadazo, enemyHP: enemy2HP);
                enemyAlive = checkStatus(enemyName: enemy2Name, attackSelected: espadazo, enemyHP: enemy2HP);
                heroAlive = enemyAttack();
            }
            break;

        case 3:
            if (puntosDeMagia >= costeBola) {
                if (enemySelection() == 1) {
                    heroAttack(attackSelected: bolaFuego, enemyName: enemy1Name, danoAtaque: danoBolaFuego, enemyHP: enemy1HP);
                    enemyAlive = checkStatus(enemyName: enemy1Name, attackSelected: bolaFuego, enemyHP: enemy1HP);
                    manaCheck(& costeMana: costeBola);
                    heroAlive = enemyAttack();
                }
                else {
                    heroAttack(attackSelected: bolaFuego, enemyName: enemy2Name, danoAtaque: danoBolaFuego, enemyHP: enemy2HP);
                    enemyAlive = checkStatus(enemyName: enemy2Name, attackSelected: bolaFuego, enemyHP: enemy2HP);
                    manaCheck(& costeMana: costeBola);
                    heroAlive = enemyAttack();
                }
            }
            else {
                cout << "No dispones de los puntos de magia necesarios para este ataque.\n";
            }
            break;
```

Hemos creado la función checkStatus, que sirve para comprobar si esta vivo un enemigo o no. Es un booleano que devuelve true o false y esta se equipara a la variable enemyAlive.




```
bool checkStatus(string enemyName, string attackSelected, int enemyHP) {
    if (enemyHP <= 0) {
        cout << "El enemigo " << enemyName << " ha sido brutalmente humillado de " << attackSelected << " y ha fallecido.\n";
        enemyHP = 0;
        return false;
    }
    else {
        cout << "Al enemigo " << enemyName << " le quedan " << enemyHP << " puntos de vida.\n";
        return true;
    }
```

He usado los parámetros por referencia para que pueda usar la misma función en diferentes ataques o enemigos.

También he creado la función heroAttack usando también los parámetros por referencia para que, de la misma manera, pueda ser utilizado en todos los ataques y enemigos

```
static void heroAttack(string attackSelected, string enemyName, int danoAtaque, int enemyHP ) {
    cout << "Nuestro heroe " << heroName << " ha atizado a " << enemyName << " con " << attackSelected << ", provocando la friolera de " << danoAtaque << " puntos de dano sobre el enemigo " << enemyName << "\n
    enemyHP = enemyHP - danoAtaque;
}
```

He añadido otra función mas que es manaCheck para que después de usar cada ataque mágico se muestre por pantalla la cantidad de puntos de magia que gasta y los que quedan.

```
static void manaCheck(int& costeMana) {
    puntosDeMagia = puntosDeMagia - costeMana;
    cout << "Este ataque te ha consumido " << costeMana << " puntos de magia. Te quedan " << puntosDeMag
}
```

Además, se han realizado cambios menores en las variables creadas al principio del código, como el aumento de la vida del héroe, para poder hacer combates más largos, así como el añadir variables String de los nombres de los ataques, con el fin de poder imprimir los nombres de dichos ataques en la función del ataque del héroe sin tener que escribir cada posibilidad de ataque y simplemente ir variando los parámetros.

```
//Atributos del Enemigo
using namespace std;
int enemy1HP = 1000;
int enemyDamage = 0;
string enemy1Name = "Malo";
bool enemyAlive = true;

//Atributos Enemigo2
int enemy2HP = 900;
int enemy2Damage;
string enemy2Name = "Malvado";
bool enemy2Alive = true;

//Atributos del Héroe
string heroName;
int heroHP = 2500;
bool heroAlive = true;
int puntosDeMagia = 1500;
int costeTormenta = 450;
int costeBola = 600;
int danoTormenta = 350;
int danoPatadon = 250;
int danoEspadazo = 400;
int danoBolaFuego = 400;
int ataqueUsado;
string enemySelected;
string bolaFuego = " una bola de fuego ";
string patadon = " un patadon ";
string espadazo = " un espadazo ";
string tormenta = " una tormenta magica ";
```

En definitiva, estos cambios han sido útiles para poder hacer el main del código bastante más limpio, usandolo únicamente para llamar a las funciones correspondientes en cada ataque y cada enemigo.

Víctor Masó

1º DAM