PROYECTO PARCIAL 1

Carlos Omar Barajas Barragán – barajas

Yeshua Miranda Morales – yeshua

Disclaimer: Al pasar el código de VScode a Word en formato de texto hubo problemas con algunas identaciones, pero dentro del código se mantiene un mejor orden al que se ve dentro de este documeno.

= Librerías =

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
```

<stdio.h> : La libreria por defecto que usamos para las funciones: getchar(), printf(), scanf().

<stdlib.h> : La usamos para las funciones rand() y srand(), donde la usamos para generar los numero aleatorios y la funcion system() que usamos para borrar la terminal

<string.h> : La usamos para el strcpy, donde asignamos los nombres de las magias

<time.h> : La usamos para generar los numeros random diferentes cada vez que volvemos a compilar y correr el codigo.

= Estructura Pokémon =

```c
typedef struct pokemon{
    char name[20];
    int hp;
    int id;
```

```
    int dormir;
    int conta_dormir;
    int defensa;
    int conta_defensa;
    int veneno;
    int conta_veneno;
    int regeneracion;
    int conta_regeneracion;
    int furia;
    int conta_furia;
    void (*action[4])(void * pokemon_jugador, void * pokemon_enemigo);
}pokemon;
```

Dentro de esta estructura a partir de la cual vamos a crear nuestros pokemones a utilizar se crearon todos los aspectos y variables necesarias para que las acciones funcionen de manera correcta, incluyendo a estas 4 acciones posibles dentro de un arreglo void de tamaño 4 que recibe 2 void pointers

= Funciones =

```
char* mostrar_turno(int turno){
    if (turno %2 == 0){
        return "Rival";
    }
    else{
        return "Jugador";
    }
}
```

Esta función mostrar_turno nos ayuda a poder indicar de manera correcta en pantalla cual de los 2 jugadores está realizando alguna acción en asignando el turno que trabaja en base al id de cada Pokémon.

```
int diferenciador (int definir_turno){
    if (definir_turno == 0){
```

```
        return 1;
    }
    else{
        return 0;
    }
}
```

La función diferenciador es utilizada para que el programa pueda alternar entre 0 y 1 el id en base al turno actual para acceder a los datos del oponente y mostrarlos en pantalla o alterarlos de manera correcta.

```
void attack(void * pokemon_jugador,void * pokemon_enemigo){

    struct pokemon *jugador = (struct pokemon *)pokemon_jugador;
    struct pokemon *enemigo = (struct pokemon *)pokemon_enemigo;

    if (enemigo->defensa == 1)
    {
        enemigo->hp-=5;
    } else if (jugador->furia ==1)
    {
        enemigo->hp-=15;
        printf("%s (%s) esta furioso y hara mas danio \n", jugador->name,
        mostrar_turno(diferenciador(jugador->id)));
        printf("\n");
    }else{
        enemigo->hp-=10;
    }
      printf("%s (%s) ataco a %s (%s)\n", jugador->name,
mostrar_turno(diferenciador(jugador->id)), enemigo->name,
mostrar_turno(diferenciador(enemigo->id)));
    printf("%s (%s) tiene ahora %d HP\n", enemigo->name,
mostrar_turno(diferenciador(enemigo->id)), enemigo->hp);
}
```

La primera de las acciones disponibles a realizar de cualquier Pokémon trabaja con 2 void pointers, uno que indica el atacante, y otro que indica el afectado, se hace una asignación que nos permite regresar al tipo de dato esperado del void pointer (lo convierte a Pokémon pointer).

Posteriormente realiza un análisis en caso de que el enemigo tenga activada su defensa o que el jugador tenga activada la furia para realizar una modificación en la salud a bajar de acuerdo a sus indicadores, de manera normal realiza un daño de 10 hp a su rival y lo muestra en pantalla

```
void block(void * pokemon_jugador,void * pokemon_enemigo){

    struct pokemon *jugador = (struct pokemon *)pokemon_jugador;
    struct pokemon *enemigo = (struct pokemon *)pokemon_enemigo;

    printf("%s (%s) uso defensa ferrea, %s aumento su defensa \n", jugador-
>name, mostrar_turno(diferenciador(jugador->id)), jugador->name);
    jugador->defensa=1;
    jugador->conta_defensa=2;

}
```

La segunda acción disponible para todos los pokemones trabaja de manera similar al ataque trabajando con punteros ya asignados a pokemon pointers, en caso de activarlo enciende un indicador de defensa y un contador que serían los turnos con los que cuenta con la defensa activada, en caso de tenerla activa, el daño de ataque se reduce a 5hp.

```
void veneno(void * pokemon_jugador,void * pokemon_enemigo){

    struct pokemon *jugador = (struct pokemon *)pokemon_jugador;
    struct pokemon *enemigo = (struct pokemon *)pokemon_enemigo;

    printf("%s (%s) uso polvo veneno en %s (%s), ha sido envenenado \n",
jugador->name, mostrar_turno(diferenciador(jugador->id)), enemigo-
>name,mostrar_turno(diferenciador(enemigo->id)));
    enemigo->veneno=1;
    enemigo->conta_veneno=3; // Dura 3 turnos el veneno

}
```

Veneno siendo una de las opciones de magia disponibles trabaja igual con 2 void pointers, el cual al invocarlo activa el indicador de veneno y un contador de 3, en

caso de haber sido envenenado se restara 5 de hp por cada turno que esté activado

```
void dormir(void * pokemon_jugador,void * pokemon_enemigo){

    struct pokemon *jugador = (struct pokemon *)pokemon_jugador;
    struct pokemon *enemigo = (struct pokemon *)pokemon_enemigo;

    printf("%s (%s) uso bostezo en %s (%s), ha caido dormido \n", jugador-
>name, mostrar_turno(diferenciador(jugador->id)), enemigo-
>name,mostrar_turno(diferenciador(enemigo->id)));
    enemigo->dormir=1;
    enemigo->conta_dormir=2;// Duerme 2 turnos
}
```

La siguiente magia disponible dormir trabaja con 2 void pointers donde al activarlo se omiten 2 turnos del jugador afectado

```
void regen(void * pokemon_jugador,void * pokemon_enemigo){

    struct pokemon *jugador = (struct pokemon *)pokemon_jugador;
    struct pokemon *enemigo = (struct pokemon *)pokemon_enemigo;

    printf("%s (%s) uso arraigo,%s (%s) empezara a recuperar hp \n",
jugador->name, mostrar_turno(diferenciador(jugador->id)), jugador-
>name,mostrar_turno(diferenciador(jugador->id)));
    jugador->regeneracion =1;
    jugador->conta_regeneracion=3;
}
```

La siguiente magia disponible, regeneración trabaja con 2 void pointer donde se activan los parámetros de inicio de la función y un contador, en caso de tener activado regeneración el jugador que lo tenga activo se le sumara 5 de hp por cada turno.

```
void furia(void * pokemon_jugador,void * pokemon_enemigo){

    struct pokemon *jugador = (struct pokemon *)pokemon_jugador;
    struct pokemon *enemigo = (struct pokemon *)pokemon_enemigo;
```

```
    printf("%s (%s) uso furia, %s esta enojado , su ataque se ve
incrementado \n", jugador->name, mostrar_turno(diferenciador(jugador->id)),
jugador->name);
    jugador->furia=1;
    jugador->conta_furia=4; // Dura 3 turnos la furia

}
```

La siguiente magia, furia funciona a partir de 2 void pointers a partir de los cuales activa los parámetros que hacen que funcione esta acción, en caso de tener activado furia, el que lo utilice realizará 15 hp de daño en lugar de 10 al utilizar ataque durante 3 turnos

```
void suerte(void * pokemon_jugador,void * pokemon_enemigo){

    struct pokemon *jugador = (struct pokemon *)pokemon_jugador;
    struct pokemon *enemigo = (struct pokemon *)pokemon_enemigo;
    int num_random= rand()%11;
    int num_escogido=-1;
    if (jugador->id == 0)
    {
        printf("%s (%s) Quiere probar su suerte \n", jugador->name,
mostrar_turno(diferenciador(jugador->id)));
        printf("Elige un numero entre el 0 y 10 , si lo adivinas pasara algo
bueno\n");
        scanf("%d",&num_escogido);
        if (num_escogido == num_random)
        {
            printf("Estas de suerte, tu pokemon se recupero al maximo\n");
            jugador->hp=100;
        }else
        {
            printf("Mala suerte el numero era %d, suerte a la
proxima\n",num_random);
        }
    }else{
        printf("%s (%s) Quiere probar su suerte \n", jugador->name,
mostrar_turno(diferenciador(jugador->id)));
        printf("Elige un numero entre el 0 y 10 , si lo adivinas pasara algo
bueno\n");
        num_escogido = rand()%11;
        if (num_escogido == num_random)
```

```
        {
            printf("Estas de suerte, tu pokemon se recupero al maximo\n");
            jugador->hp=100;
        }else
        {
            printf("Mala suerte el numero era %d, suerte a la
proxima\n",num_random);
        }
        getchar();

    }

}
```

La ultima magia, suerte funciona como un adivinador de números random entre el 0 y el 10 y en caso de llegar a adivinarlo le restaura todo el hp a 100

```
void magia_set(pokemon* battle_ptr, void (**magia_array)(void*, void*),
char** magia_names, int contador) {


    int magia_anterior = -1;
    int magia_escogida;

    if ((battle_ptr + contador)->id == 0) { // jugador
        // LLena las magias
        for (int j = 2; j < 4; j++) {
            // Bucle para pedir una opción hasta que sea válida.
            do {
                printf("\n--- Escoge la Magia para el espacio %d ---\n", j);
                for (int i = 0; i < 5; i++) {
                    printf("%d. %s\n", i, *(magia_names + i));
                }

                printf("Jugador, escoge tu magia: ");
                scanf("%d", &magia_escogida);
                fflush(stdout);

                if (magia_escogida == magia_anterior) {
                    printf("Ya has escogido esa magia. Elige otra.\n");
```

```
                    while (getchar() != '\n'); // Se limpia el buffer
                }
            } while (magia_escogida == magia_anterior);

            (battle_ptr + contador)->action[j] =
magia_array[magia_escogida];

            magia_anterior = magia_escogida;
            printf("\n");
            printf("Magia '%s' asignada con exito.\n", *(magia_names +
magia_escogida));
        }
    }
    else{
    // Bot elige magia
    printf("---Magia escogida por el bot---\n");
    printf("\n");

       for (int j = 2; j < 4; j++) {

            do {
               magia_escogida = rand()%5;
            } while (magia_escogida == magia_anterior);

            (battle_ptr + contador)->action[j] =
magia_array[magia_escogida];

            magia_anterior = magia_escogida;
            printf("Magia '%s' asignada con exito.\n", *(magia_names +
magia_escogida));
        }
    }
}
```

Dentro de esta función se hace la asignación de las magias a cada uno de los participantes de la partida, se hacen ciertas validaciones para que no se repitan magias dentro de sus acciones así como asignarlas a distintas posiciones de sus ataques

```
void pokemon_set(pokemon * battle_ptr,pokemon * pokedex_pointer, void
(**magia_array)(void*, void*), char **magia_names){
```

```
    int pokemon_escogido=-1;
    pokemon * pokemon_nombres = pokedex_pointer;

    for (int i = 0; i < 5; i++,pokemon_nombres++)
    {
        printf("[%d]-- %s \n",i,(pokemon_nombres->name));
    }


    while (pokemon_escogido <0 || pokemon_escogido >= 5)
    {
        printf("Jugador escoge tu pokemon\n");
        scanf("%d", &pokemon_escogido);
        fflush(stdout);
    }


    for (int i= 0; i<2; i++){//3

        *(battle_ptr+i) = *(pokedex_pointer+pokemon_escogido);
        (battle_ptr+i)->id = i;

        magia_set(battle_ptr,magia_array,magia_names,i);
        printf("\n");

        pokemon_escogido = rand()%5;
    }

}
```

En esta función de pokemon_set se reciben varios parámetros para dentro de ella misma tambien poder utilizar magia_set, dentro de ella se muestra el pokedex que incluye a todos los pokemones que se pueden escoger para la batalla, posteriormente se hace una división para determinar si se debe de ingresar por el usuario o de manera aleatoria por la computadora a través de un random number, una vez realizada la elección asignarlo a la estructura pokemon dentro de su apartado de acciones en la tercera y cuarta posición del arreglo.

```
void pokemon_preview(pokemon * battle_ptr){
```

```
    printf("Preview pokemons:\n");

    printf("Pokemon Aliado: %s\n",(battle_ptr)->name);
    printf("Salud propia: %d\n",(battle_ptr)->hp);


    printf("Pokemon enemigo: %s\n",(battle_ptr+1)->name);
    printf("Salud enemigo: %d\n",(battle_ptr+1)->hp);


   printf("--------------------------------\n");
   printf("\n");
}
```

La función pokemon_preview se utiliza para poder mostrar durante toda la partida sus datos como nuestro pokemon y sus salud, así como el pokemon rival y su salud

```
void limpiar_pantalla(){
    system("cls");
}
```

La función limpiar_pantalla se usa a lo largo del código para hacer que la terminal donde se corre el programa se vea mas limpia y se eliminen datos innecesarios de turnos anteriores

```
void presentacion(){
    printf("-----PokeBattle-----\n");
    printf("Presiona enter para continuar y empezar a pelear\n");
    getchar();
}
```

Dentro de esta función se hace la presentación inicial de la partida

```
void final(void * pokemon_jugador,void * pokemon_enemigo){
```

```
    struct pokemon *jugador = (struct pokemon *)pokemon_jugador;
    struct pokemon *enemigo = (struct pokemon *)pokemon_enemigo;

    if (jugador->hp <= 0)
    {
        printf("%s (%s) se ha debilidato, %s (%s) gana la batalla \n",
jugador->name, mostrar_turno(diferenciador(jugador->id)), enemigo-
>name,mostrar_turno(diferenciador(enemigo->id)));
        printf("---Perdiste---\n");
        getchar();
    }else{
        printf("%s (%s) se ha debilidato, %s (%s) gana la batalla \n",
enemigo->name, mostrar_turno(diferenciador(enemigo->id)), jugador-
>name,mostrar_turno(diferenciador(jugador->id)));
        printf("---Ganaste---\n");
        getchar();
    }




}
```

Dentro de este apartado se analiza la vida de los oponentes y en caso de haber llegado a 0 se muestra al vencedor de la partida

Nombre_magia

```
char * nombre_magia(pokemon * battle_ptr) {
    void (*magia_actual)(void*, void*) = *(battle_ptr->action + 2);

    // Comparamos ese puntero con las funciones de magia que conocemos
    if (magia_actual == veneno) {
        return "Veneno"; // Si es la función 'veneno', devolvemos el texto
"Veneno"
    }
    else if (magia_actual == dormir) {
        return "Dormir";
```

```
    }
    else if(magia_actual == regen){
        return "Regeneracion";
    }else if (magia_actual == furia)
    {
        return "Furia";
    }else if (magia_actual == suerte)
    {
        return "Suerte";
    }
    return NULL;
}
```

La función nombre_magia lo que hace es asignar el nombre correcto dentro de las impresiones de magia de acuerdo a lo seleccionado y donde se debe de almacenar

= Main =

```
int main(){

    srand(time(NULL));



   pokemon Pikachu = {
        .name = "Pikachu",
        .defensa = 0,
        .conta_defensa = 0,
        .dormir = 0,
        .conta_dormir =0,
        .veneno = 0,
        .conta_veneno =0,
        .regeneracion =0,
        .conta_regeneracion=0,
        .furia=0,
        .conta_furia=0,
        .hp = 100,
        .action = { attack, block,NULL,NULL}
    };
```

```
pokemon Charmander = {
    .name = "Charmander",
    .defensa = 0,
    .conta_defensa = 0,
    .dormir = 0,
    .conta_dormir =0,
    .veneno = 0,
    .conta_veneno =0,
    .regeneracion =0,
    .conta_regeneracion=0,
    .furia=0,
    .conta_furia=0,
    .hp = 100,
    .action = { attack, block,NULL,NULL}
};

pokemon Squirtle = {
    .name = "Squirtle",
    .defensa = 0,
    .conta_defensa = 0,
    .dormir = 0,
    .conta_dormir =0,
    .veneno = 0,
    .conta_veneno =0,
    .regeneracion =0,
    .conta_regeneracion=0,
    .furia=0,
    .conta_furia=0,
    .hp = 100,
    .action = { attack, block,NULL,NULL}
};

pokemon Bulbasur = {
    .name = "Bulbasur",
    .defensa = 0,
    .conta_defensa = 0,
    .dormir = 0,
    .conta_dormir =0,
    .veneno = 0,
    .conta_veneno =0,
    .regeneracion =0,
    .conta_regeneracion=0,
    .furia=0,
    .conta_furia=0,
```

```
        .hp = 100,
        .action = { attack, block,NULL,NULL}
    };

    pokemon Mewtwo = {
        .name = "Mewtwo",
        .defensa = 0,
        .conta_defensa = 0,
        .dormir = 0,
        .conta_dormir =0,
        .veneno = 0,
        .conta_veneno =0,
        .regeneracion =0,
        .conta_regeneracion=0,
        .furia=0,
        .conta_furia=0,
        .hp = 100,
        .action = { attack, block,NULL,NULL}
    };


    pokemon battle [2];
    pokemon * battle_ptr = battle;
```

Se inician creando cada uno de los pokemones con los parámetros necesarios, además de crear una batalla pokemon de tamaño 2 y su apuntador

```
    void (*magia_array[5])(void*, void*) = {veneno, dormir, regen, furia,
suerte};
    char magia_1[30] ="Veneno";
    char magia_2[30] = "Dormir";
    char magia_3[30] = "Regeneracion";
    char magia_4[30] = "Furia";
    char magia_5[30] = "Suerte";
    char *nombres_magia[]={magia_1,magia_2,magia_3,magia_4,magia_5};
    char **ptr_magia=nombres_magia;

   pokemon pokedex [5]={Pikachu,Charmander,Squirtle,Bulbasur,Mewtwo};
   pokemon * pokedex_ptr = pokedex;
```

Se declaran los arreglos que almacenan al pokedex y a las opciones de magias

```
// Presentacion

    presentacion();

   // Establece los pokemones
    pokemon_set(battle_ptr,pokedex_ptr, magia_array,ptr_magia);

       // Declara nombres ataques
    char atq1[30] ="Ataque";
    char atq2[30] = "Defensa";
    char atq3[30] = ".";
    char atq4[30] = ".";
    char *nombres_ataques[]={atq1,atq2,atq3,atq4};
    char **ptr_nombres=nombres_ataques;
    int constante =1, turno = 1, opcion_atque =-1;
```

Se inicia la presentación de la partida además de invocar la función que inicia a crear la partida inicializando los pokemones tanto para el jugadorcomo el CPU, se declaran los ataques a imprimir

```
    // Solo muestra los pokemones
   pokemon_preview(battle_ptr);
   printf("Presiona enter para continuar\n");
   getchar();
   getchar();
   limpiar_pantalla();

   // Jugador
```

Se inicia a mostrar los datos de la partida previo a iniciarla

```
    while (battle[0].hp>0 && battle[1].hp>0){
```

```
        // Hacer que se limpie la terminal, que se vea bonito
       limpiar_pantalla();

        int definir_turno = (turno%2);

        printf("Detalles de la partida: \n");

        printf("(Jugador) %s Salud : %d\n",(battle)->name, (battle)->hp);

        printf("(Enemigo) %s Salud : %d\n",(battle+1)->name, (battle+1)-
>hp);

        ptr_nombres=nombres_ataques;
```

A partir de este punto se inicia a almacenar todo en el while para marcar todo el transcurso de la partida, se inicia a tener un determinante que va a variar entre los turnos (definir_turno), y se muestran los datos actuales de cada jugador en cada turno

```
        if (definir_turno == 1){

            printf("\n");
            printf("Turno (%s)\n",mostrar_turno(definir_turno));

            // Revisamos dormir
            if (battle->dormir ==1){
                battle->conta_dormir--;
                printf("%s (%s) esta profundaente dormido, %s pierde el
turno \n",battle->name,mostrar_turno(definir_turno),battle->name);
                getchar();
                if (battle->conta_dormir<= 0){
                    printf("%s (%s) se ha despertado, a vuelto a la
normalidad\n",battle->name,mostrar_turno(definir_turno));
                    battle->dormir =0;
                    getchar();
                }
            } else{

                printf("(%s) elige el ataque de tu
%s:\n",mostrar_turno(definir_turno),(battle_ptr)->name);
                strcpy(atq3, nombre_magia(battle_ptr,2));
                strcpy(atq4, nombre_magia(battle_ptr,3));
```

```
                for (int i = 0; i < 4; i++,ptr_nombres++)
                {
                    printf("%d. %s\n",i,*ptr_nombres);
                }
                scanf("%d",&opcion_atque);
                printf("\n");
                battle[diferenciador(definir_turno)].action[opcion_atque]((b
attle_ptr+diferenciador(definir_turno)),(battle_ptr+definir_turno));

                printf("Presiona enter para continuar\n");
                getchar();
                getchar();
                printf("Estados activos: \n");
```

Se inicia con el turno del jugador donde se inicia con el identificador de dormir para que no se ejecute su acción en caso de haber sido congelado previamente, en caso de no haber sido congelado se pasa a que el jugador escoga su ataque y se realizan distintos análisis

```
                // Evaluamos defensa
                if (battle->defensa ==1)
                {
                    battle->conta_defensa--;
                    if (battle->conta_defensa <= 0)
                    {
                        printf("La defensa de %s (%s) a vuelto a la
normalidad\n",battle->name,mostrar_turno(definir_turno));
                        battle->defensa=0;
                        getchar();
                    }
                }
```

Se evalúa el identificador de defensa para ir disminuyéndolo o mostrar que ya no se tiene activada la defensa

```
// Evaluamos veneno
                if (battle->veneno ==1)
                {
```

```
                    battle->conta_veneno--;
                    printf("%s (%s) se resiste al veneno, %s pierde 5hp
\n",battle->name,mostrar_turno(definir_turno),battle->name);
                    getchar();
                    battle->hp-=5;
                    if (battle->conta_veneno <= 0)
                    {
                        printf("%s (%s) se ha curado del veneno, a vuelto a
la normalidad\n",battle->name,mostrar_turno(definir_turno));
                        battle->veneno =0;
                    }
                }
```

Posteriormente se analiza si se cuenta si el jugador actual se encuentra envenenado y se realiza la lógica del ataque a lo largo de los turnos y en caso de no tenerlo se sigue con el resto del análisis

```
 if (battle->regeneracion ==1)
                {
                    battle->conta_regeneracion--;
                    printf("%s (%s) recupera 5hp \n",battle-
>name,mostrar_turno(definir_turno));
                    getchar();
                    battle->hp+=5;
                    if (battle->conta_regeneracion <= 0)
                    {
                        printf("%s (%s) ha perdido la regeneracion, a vuelto
a la normalidad\n",battle->name,mostrar_turno(definir_turno));
                        battle->regeneracion =0;
                        getchar();
                    }
                }
```

Se analiza si se tiene el identificador de regeneración así como su contador y en caso de tenerlo activado se va regenerando

```
                if (battle->furia ==1)
                {
```

```
                    battle->conta_furia--;
                    printf("%s (%s) Sigue enojado \n",battle-
>name,mostrar_turno(definir_turno));
                    getchar();
                    if (battle->conta_furia <= 0)
                    {
                        printf("%s (%s) ha calmado, a vuelto a la
normalidad\n",battle->name,mostrar_turno(definir_turno));
                        battle->furia =0;
                        getchar();
                    }
                }
            }
```

Se analiza el ultimo parámetro de furio y se sigue la lógica del ataque

```
        else if (definir_turno == 0){

            printf("\n");
            printf("Turno (%s)\n",mostrar_turno(definir_turno));

            // Revisamos si duerme
            if ((battle+1)->dormir ==1){
                (battle+1)->conta_dormir--;
                printf("%s (%s) esta profundaente dormido, %s pierde el
turno \n",(battle+1)->name,mostrar_turno(definir_turno),(battle+1)->name);
                getchar();
                if ((battle+1)->conta_dormir<= 0){
                    printf("%s (%s) se ha despertado, a vuelto a la
normalidad\n",(battle+1)->name,mostrar_turno(definir_turno));
                    (battle+1)->dormir =0;
                    getchar();
                }
            }else{
                printf("(%s) eligiendo el ataque de su
%s\n",mostrar_turno(definir_turno),(battle_ptr+1)->name);
                printf("Enemigo pensando\n");
                printf("Presiona enter para continuar\n");

                //opcion_atque= 2; //Hace solo la magia
                opcion_atque = rand()%5;// elige del 0 al 4
```

```
                getchar();



                battle[diferenciador(definir_turno)].action[opcion_atque]((b
attle_ptr+diferenciador(definir_turno)),(battle_ptr+definir_turno));

                printf("Presiona enter para continuar\n");
                getchar();
                printf("Estados activos: \n");

                // Revisamos defensa
                if ((battle+1)->defensa ==1)
                {
                    (battle+1)->conta_defensa--;
                    if ((battle+1)->conta_defensa <= 0)
                    {
                        printf("La defensa de %s (%s) a vuelto a la
normalidad\n",(battle+1)->name,mostrar_turno(definir_turno));
                        (battle+1)->defensa=0;
                        getchar();
                    }
                }
                // Revisamos veneno
                if ((battle+1)->veneno ==1)
                {
                    (battle+1)->conta_veneno--;
                    printf("%s (%s) se resiste al veneno, %s pierde 5hp
\n",(battle+1)->name,mostrar_turno(definir_turno),(battle+1)->name);
                    getchar();
                    (battle+1)->hp-=5;
                    if ((battle+1)->conta_veneno <= 0)
                    {
                        printf("%s (%s) se ha curado del veneno, a vuelto a
la normalidad\n",(battle+1)->name,mostrar_turno(definir_turno));
                        (battle+1)->veneno =0;
                        getchar();
                    }
                }

                // Evaluamos regeneracion
                if ((battle+1)->regeneracion ==1)
                {
                    (battle+1)->conta_regeneracion--;
```

```
                    printf("%s (%s) recupera 5hp \n",(battle+1)-
>name,mostrar_turno(definir_turno));
                    getchar();
                    (battle+1)->hp+=5;
                    if ((battle+1)->conta_regeneracion <= 0)
                    {
                        printf("%s (%s) ha perdido la regeneracion, a vuelto
a la normalidad\n",(battle+1)->name,mostrar_turno(definir_turno));
                        (battle+1)->regeneracion =0;
                        getchar();
                    }
                }
                // Evaluamos furia
                if ((battle+1)->furia ==1)
                {
                    (battle+1)->conta_furia--;
                    printf("%s (%s) Sigue enojado \n",(battle+1)-
>name,mostrar_turno(definir_turno));
                    getchar();
                    if ((battle+1)->conta_furia <= 0)
                    {
                        printf("%s (%s) ha calmado, a vuelto a la
normalidad\n",(battle+1)->name,mostrar_turno(definir_turno));
                        (battle+1)->furia =0;
                        getchar();
                    }
                }

            }

        }
        turno+=1;
        printf("\n");
    }
    final(battle_ptr,battle_ptr+1);
}
```

Por ultimo se aplican las mismas revisiones en el caso del CPU y se desarrolla la partida hasta que alguno de los 2 jugadores llegue a tener <=0 de vida para mostrar por ultimo la pantalla final de la partida