En conclusión, este fue un proyecto complejo de implementar. Sobre la lógica del juego. Decidimos en un inicio implementar el proyecto de forma gráfica, por lo cual el equipo se separó en dos. Los encargados de implementar la parte gráfica y los que íbamos a implementar la lógica del juego.

Para la parte lógica, se implementaron en un inicio las estructuras de Pokemon y de Stats:

struct pokemon  
{  
 *Type* type1;  
 *Type* type2;  
 double \*weaknesses;  
 *Stats* \*stats;  
 char \*name;  
 *List* \*attacks;  
 *State* current\_state;  
 int alive;  
};

typedef struct stats  
{  
 int level;  
  
 *// Base stats* int base\_hp;  
 int base\_attack;  
 int base\_defense;  
 int base\_speed;  
 int base\_S\_attack;  
 int base\_S\_defense;  
  
 *// Variable stats* int variable\_hp;  
 int variable\_attack;  
 int variable\_defense;  
 int variable\_speed;  
 int variable\_S\_attack;  
 int variable\_S\_defense;  
  
 *// Stats* int fixed\_hp;  
 int hp;  
 int attack;  
 int defense;  
 int speed;  
 int S\_attack;  
 int S\_defense;  
}*Stats*;

Las cuales se mantuvieron prácticamente de la misma forma a excepción de las variables “fixed\_hp” y “alive”. Se utilizaron funciones como “new\_pokemon” para inicializar nuestras estructuras de pokemons.

*Pokemon*\* new\_pokemon(char \**name*, *Type type1*, *Type type2*, int *hp*, int *attack*, int *defense*, int *speed*, int *S\_attack*, int *S\_defense*)  
{  
*// Variables important for the rand function and srand  
 time\_t* t;  
  
 *Pokemon* \*pokemon\_new = malloc(sizeof (*Pokemon*));  
 pokemon\_new->stats = malloc(sizeof (*Stats*));  
 pokemon\_new->stats->level = 1;  
 pokemon\_new->alive = 1;  
 pokemon\_new->type1 = *type1*;  
 pokemon\_new->type2 = *type2*;  
  
*// Setting varible values* srand((unsigned) time(&t));  
 pokemon\_new->stats->variable\_hp = (rand() % 15) + 1;  
  
 srand((unsigned) time(&t));  
 pokemon\_new->stats->variable\_attack = (rand() % 15) + 1;  
  
 srand((unsigned) time(&t));  
 pokemon\_new->stats->variable\_defense = (rand() % 15) + 1;  
  
 srand((unsigned) time(&t));  
 pokemon\_new->stats->variable\_speed = (rand() % 15) + 1;  
  
 srand((unsigned) time(&t));  
 pokemon\_new->stats->variable\_S\_attack = (rand() % 15) + 1;  
  
 srand((unsigned) time(&t));  
 pokemon\_new->stats->variable\_S\_defense = (rand() % 15) + 1;  
  
  
*// Set the base values* pokemon\_new->stats->base\_hp = *hp*;  
 pokemon\_new->stats->base\_attack = *attack*;  
 pokemon\_new->stats->base\_defense = *defense*;  
 pokemon\_new->stats->base\_speed = *speed*;  
 pokemon\_new->stats->base\_S\_attack = *S\_attack*;  
 pokemon\_new->stats->base\_S\_defense = *S\_defense*;  
  
 *// Set the hp* pokemon\_new->stats->hp = get\_stat(1, *hp*, pokemon\_new->stats->variable\_hp, 1);  
  
 *// Set the attack* pokemon\_new->stats->attack = get\_stat(1, *attack*, pokemon\_new->stats->variable\_attack, 0);  
  
 *// Set the defense* pokemon\_new->stats->defense = get\_stat(1, *defense*, pokemon\_new->stats->variable\_defense, 0);  
  
 *// Set the speed* pokemon\_new->stats->speed = get\_stat(1, *speed*, pokemon\_new->stats->variable\_speed, 0);  
  
 *// Set the Special attack* pokemon\_new->stats->S\_attack = get\_stat(1, *S\_attack*, pokemon\_new->stats->variable\_S\_attack, 0);  
  
 *// Set the Special defense* pokemon\_new->stats->S\_defense = get\_stat(1, *S\_defense*, pokemon\_new->stats->variable\_S\_defense, 0);  
  
 pokemon\_new->stats->fixed\_hp = pokemon\_new->stats->hp;  
  
 pokemon\_new->name = *name*;  
  
 pokemon\_new->current\_state = normal\_state;  
  
 pokemon\_new->attacks = list\_new();  
  
 pokemon\_new->weaknesses = get\_weakness(*type1*, *type2*);  
  
 return pokemon\_new;  
}

Prácticamente el juego implementó las funciones de pelea, de modificación de estado, entre otras, antes de integrarlas con la parte gráfica. Sin embargo, al no poder hacer finalmente esa integración, se tomó la decisión de desechar todo lo gráfico. Por lo cual, al inicio del juego, se le pregunta al jugar, por medio de línea de comandos, si desea iniciar un juego nuevo. Debido, al tiempo de desarrollo del proyecto y a la magnitud de este, el juego no cuenta con una función de guardar y cargar la partida.

Al iniciar la partida, se corre la función de “init\_pokemons()”, la cual crea un archivo en modo escritura, crea 20 nuevos pokemons y los escribe dentro de un archivo de forma binaria. Durante la implementación de la función, tuvimos ciertos problemas al escribir todos los 20 pokemons en el archivo, por lo cual, decidimos abrir el archivo, escribir en él, y cerrarlo. Así del mismo modo hasta escribir los 20 pokemons.

void init\_pokemons()  
{  
 *FILE* \*file;  
 file = fopen("../pokemon.txt", "w");  
  
 *Pokemon* \*p1 = new\_pokemon("Bulbasaur",grass\_type,poison\_type,45,49,49,45,65,65);  
 define\_attacks("Razor Leaf", p1,grass\_type,phisical,55,95,25,normal\_state,0,0,0,0);  
 define\_attacks("Growth",p1,normal\_type,stat,0,-100,40,normal\_state,100,S\_attack\_affected\_stat,1,1);  
 define\_attacks("Body slam",p1,normal\_type,phisical,85,100,15,paralyzed\_state,30,0,0,0);  
 define\_attacks("Stun spore", p1,grass\_type,stat,0,75,30,paralyzed\_state,100,0,0,0);  
  
 set\_level(50, p1);  
 fwrite(p1, sizeof(*Pokemon*), 1, file);  
 fclose(file);

file = fopen("../pokemon.txt", "a");  
*Pokemon* \*p2 = new\_pokemon("Ivysaur",grass\_type,poison\_type,60,62,63,60,80,80);  
define\_attacks("Razor Leaf", p2,grass\_type,phisical,55,95,25,normal\_state,0,0,0,0);  
define\_attacks("Growth", p2,normal\_type,stat,0,-100,40,normal\_state,100,S\_attack\_affected\_stat,1,1);  
define\_attacks("Stun spore",p2, grass\_type,stat,0,75,30,paralyzed\_state,100,0,0,0);  
define\_attacks("Body slam",p2,normal\_type,phisical,85,100,15,paralyzed\_state,30,0,0,0);  
  
set\_level(50, p2);  
fwrite(p2, sizeof(*Pokemon*), 1, file);  
fclose(file);

Continuando con el juego, se le setean los pokemons al jugador por medio de la función de “init\_player()”. En esta función, decidimos recuperar los 20 pokemons del archivo y asignársele 6 de manera aleatoria al jugador. Por otro lado, a los enemigos se le setean de manera fija los jugadores.

Puesto que excluimos la parte gráfica del proyecto, decidimos agregarle diálogos e historia al juego:

printf("You are Red... The best Pokemon trainer in the world.\n");  
**JUMP**sleep(2);  
printf("You are always proud of yourself, and always wish\n");  
printf("to prove to others what you are capable of.\n");  
**JUMP**sleep(2);  
printf("Recently, you became aware of a Pokemon tournament, \n");  
printf("where you, obviously, needed to participate.\n");  
**JUMP**sleep(2);  
printf("So, you grabbed the best Pokemons you have, 5 dollars to register for the tournament,\n");  
printf("a sandwich for lunch, and directed yourself towards the tournament.\n");

Para el buen manejo de los textos en pantalla, decidimos utilizar la librería “unistd.h”, la cual tiene la función de “sleep()”. Esta función nos permitió llevar un buen manejo de tiempo de la aparición de los textos en pantalla.

El sistema de peleas del juego depende de la selección de los movimientos del jugador y del contrincante. Además de que el turno se decide por medio de determinar que pokemon es más rápido. El más rápido ataca primero:

if(get\_pokemon\_speed(player\_pokemon) > get\_pokemon\_speed(enemy\_pokemon))  
{  
 player\_move(player\_pokemon, enemy\_pokemon, players\_attack);  
 if(get\_pokemon\_alive(enemy\_pokemon))  
 {  
 player\_move(enemy\_pokemon, player\_pokemon, enemyAttack);  
 }  
} else  
{  
 player\_move(enemy\_pokemon, player\_pokemon, enemyAttack);  
 if(get\_pokemon\_alive(player\_pokemon))  
 {  
 player\_move(player\_pokemon, enemy\_pokemon, players\_attack);  
 }  
}

Existen ataques que pueden provocar estados de parálisis, dormir, congelar, quemar y envenenar, por lo cual, se checan los estados de los pokemons constantemente para modificar sus estadísticas.

Al ganar una batalla, tomamos la decisión de que todos nuestros pokemons regresen a la vida y a sus estadísticas base. También tomamos la decisión de que suban de nivel para que exista un sentimiento de haber ganado experiencia:

player\_normalize(*player*);  
player\_set\_level(*player*, get\_pokemon\_level(player\_pokemon) + 5);  
printf("YOU WON!!!\n");  
**JUMP**sleep(1);  
printf("You have defeated %s", get\_enemy\_name(*enemy*));  
**JUMP**sleep(1);  
return 1;

Por último, el jugador enfrenta a 3 contrincantes. Si gana las batallas, gana la copa del torneo:

printf("Finally, you received the tournament cup. Congratulations!!!\n");  
printf(" \_\_\_\_\_\_\_\n");  
printf(" | |\n");  
printf("(| RED |)\n");  
printf(" | #X |\n");  
printf(" \\ /\n");  
printf(" `---'\n");  
printf(" \_|\_|\_\n");

Si pierde, se termina el juego:

printf(" \_\_\_\_\_ \_\_\_\_ \n");  
printf("/ \_\_\_\_| / \_\_ \\ \n");  
printf("| | \_\_ \_\_ \_ \_ \_\_ \_\_\_ \_\_\_ | | | |\_ \_\_\_\_\_ \_ \_\_ \n");  
printf("| | |\_ |/ \_` | '\_ ` \_ \\ / \_ \\ | | | \\ \\ / / \_ \\ '\_\_|\n");  
printf("| |\_\_| | (\_| | | | | | | \_\_/ | |\_\_| |\\ V / \_\_/ | \n");  
printf(" \\\_\_\_\_\_|\\\_\_,\_|\_| |\_| |\_|\\\_\_\_| \\\_\_\_\_/ \\\_/ \\\_\_\_|\_| \n");  
**JUMP**sleep(1);  
printf("Thanks for playing... See you next time.\n");