Universidad Rafael Landivar
Facultad de Ingeniería
Ingeniería Industrial
Introducción a la programación
Sección 18
Catedrático: Herwing Alexis Rodríguez Franco

Proyecto Final

"CLASH ROYALE"

Valentina Cividanis González 1063623
César Alberto Jaimes Carrillo 1249523
César Daniel Villela Pérez 1064023

Guatemala, 15 de noviembre 2023

I. RESUMEN

En el presente proyecto final de introducción a la programació, se llevó a cabo a lo largo del curso del presente año por la sección 10, el cual trató principalmente sobre crear un código basado en el juego “Clash Royale” se definieron los 4 Decks o mazos, para que dos jugadores seleccionen 2 de ellos. Se construyó un programa en C# el cual determina cual es el mejor deck a utilizar en base a elixir, puntos de vida y puntos de daño.

Al momento de llevar a cabo dicho proyecto se realizaron 4 procedimientos diferentes. El primero fue crear un algoritmo paso a paso para elegir mazos en clash royale y una simulación 1 contra 1 y luego se construyó una lógica programática de juego mediante operadores lógicos y operadores de comparación. El segundo procedimiento fue recolectar toda la información de cada una de las cartas de cada deck en una hoja de excel. El tercer procedimiento fue realizar todo el código en C# para verificar estadísticamente (según los puntos de daño) que deck es el mejor y quien ganaría en una partida. Finalmente el último paso fué grabar un video simulando una partida en el link proveído por el catedrático y comparar el ganador con el código.

II. VIDEO DE EXPLICACION

https://www.youtube.com/watch?v=Xuh3YHS-mcU

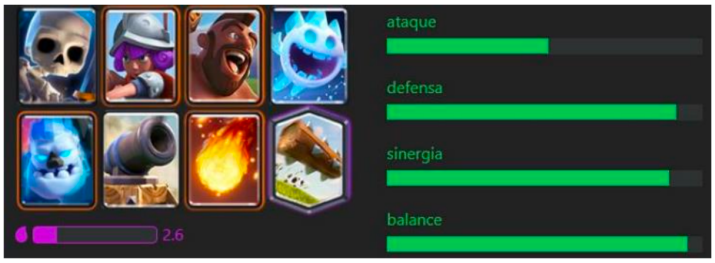
III. INTRODUCCION A LA PROGRAMACION DEL JUEGO

- Algoritmo

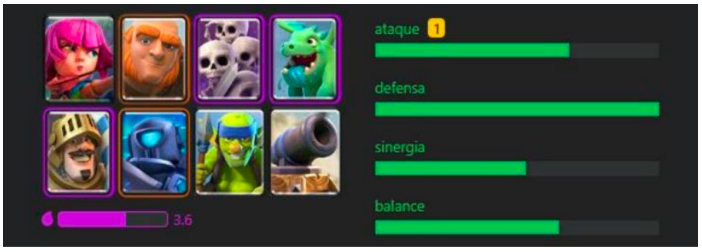
Paso 1: inicio
Paso 2: Abrir el juego
Paso 3: Ir al apartado de decks
Paso 4: Crear el deck “Hog 2.6 Cycle”
4.1: Seleccionar los esqueletos
4.2: Seleccionar `la mosquetera
4.3: Seleccionar el montapuercos
4.4: Seleccionar el espíritu de hielo
4.5: Seleccionar el gólem de hielo
4.6: Seleccionar el cañón
4.7: Seleccionar la bola de fuego
4.8: Seleccionar el tronco
Paso 5: Crear el deck “Giant mortar”
5.1: seleccionar las arqueras
5.2: seleccionar al gigante
5.3: seleccionar el ejercito de esqueletos
5.4: seleccionar el baby dragon
5.5: seleccionar el príncipe
5.6: seleccionar el mini P.E.K.K.A
5.7: seleccionar los duendes con lanza
5.8: seleccionar el mortero
Paso 6: ir al apartado de “batalla”
Paso 7: desafiar al rival
Paso 8: Jugar la batalla
Paso 9: Sumar los puntos de vida de las cartas del deck “Hog 2.6 Cycle”
9.1: Sumar los puntos de daño de las cartas del deck “Hog 2.6 Cycle”
Paso 10: Sumar los puntos de vida de las cartas del deck “Giant mortar”
10.1: Sumar los puntos de daño de las cartas del deck “Giant mortar”
Paso 11: Identificar que suma es mayor entre los puntos de vida y de daño.Paso 12: Si en puntos de vida gana “Hog 2.6 Cycle”
12.1: entonces gana “Hog 2.6 Cycle”
Paso 13: Si en puntos de daño gana “Giant mortar”
13.1: entonces gana “Giant mortar”
Paso 14: indicar que deck es el ganador
Paso 15: fin de la batalla
Paso 16: Fin

- Lógica programática

Hog 2.6 cycle



Giant Mortar



Comparación

Se puede observar que el Hog cycle tiene ventaja en lo que viene siendo la sinergia y el balance, mientras que el Giant Mortar tiene ventaja en el ataque y defensa. El Hog cycle es muy útil si el estilo de batalla es rápido y eficaz ya que al tener un ciclado rápido se puede hacer presión constante, mientras que si lo táctico es lo que buscas, sin duda el Giant Mortar es la elección correcta ya que este modo de ataque requiere de más apoyo y cuenta con dos ataques y no solo uno.

IV. VERIFICACION Y CONSTRUCCION DE MAZOS

Hog 2.6 Cycle	Carta	Daño	Vida	Velocidad	Tipo de carta	Calidad	Elixir	Coste medio elixir
	Esqueletos	81	81	Alta	Tropa	Común	1	2.6
	Mosquetera	218	720	Media	Tropa	Especial	4	
	Montapuercos	198	1696	Muy alta	Tropa	Especial	4	
	Espiritu de hielo	110	230	Muy alta	Tropa	Común	1	
	Gólem de hielo	84	1197	Baja	Tropa	Especial	2	
	Cañón	212	824	N/A	Estructura	Común	3	
	Bola de fuego	689	N/A	N/A	Hechizo	Especial	4	
	Tronco	290	N/A	N/A	Hechizo	Legendaria	2	

Giant Mortar	Carta	Daño	Vida	Velocidad	Tipo de carta	Calidad	Elixir	Coste medio elixir
	Arqueras	107	304	Media	Tropa	Común	3	3.8
	Gigante	254	4091	Baja	Tropa	Especial	5	
	Ejército de esqueletos	81	81	Alta	Tropa	Común	1	
	Bebé Dragón	160	1152	Alta	Tropa	Épica	4	
	Príncipe	392	1920	Media	Tropa	Épica	5	
	Mini P.E.K.K.A	720	1361	Alta	Tropa	Especial	4	
	Duendes con lanza	81	133	Muy alta	Tropa	Común	2	
	Mortero	266	1369	N/A	Estructura	Común	4	

XBOW 2.9	Carta	Daño	Vida	Velocidad	Tipo de carta	Calidad	Elixir	Coste medio elixir
	Arqueras	107	304	Media	Tropa	Común	3	2.9
	Esqueletos	81	81	Alta	Tropa	Común	1	
	Espiritu de hielo	110	230	Muy alta	Tropa	Común	1	
	Golém de hielo	84	1197	Baja	Tropa	Especial	2	
	Torre tesla	230	1152	0	Estructura	Común	4	
	Ballesta	41	1600	0	Estructura	Épica	6	
	Bola de fuego	689	0	0	Hechizo	Especial	4	
	Tronco	290	0	0	Hechizo	Legendaria	2	

Spell Bait	Carta	Daño	Vida	Velocidad	Tipo de carta	Calidad	Elixir	Coste medio elixir
	Caballero	202	1766	Media	Tropa	Común	3	3.3
	Princesa	169	261	Media	Tropa	Legendaria	3	
	Espiritu de hielo	110	230	Muy alta	Tropa	Común	1	
	Pandilla de duendes	81	133	Muy alta	Tropa	Común	3	
	Torre infernal	848	1749	0	Estructura	Especial	5	
	Cohete	1484	0	0	Hechizo	Especial	6	
	Barril de duendes	120	202	Muy alta	Hechizo	Épica	3	
	Tronco	290	0	0	Hechizo	Legendaria	2	

V. PROGRAMACION C#

- Clase Decks

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Proyecto_Lab_Progra
{
        internal class Decks
        {
                private string jugador;
                private string equipo;
                private string nom_carta;
                private string tipo_carta;
                private int elixir;
                private int puntos_vida;
                private int daño;
                public string[,] deck;

      public Decks()
                {
                        /*declarando el contructor como matriz para guardar la información de cada carta de cada deck*/
                        this.deck = new string[8, 5];
                }

       public void AddCard(int puntos_vida, int elixir, int daño, string nom_carta, string tipo, int posicion)
      {
                        /*se agrega la información específica de la carta*/
```

```
        deck[posicion, 0] = puntos_vida.ToString();
        deck[posicion, 1] = daño.ToString();
        deck[posicion, 2] = elixir.ToString();
        deck[posicion, 3] = nom_carta.ToString();
        deck[posicion, 4] = tipo.ToString();
    }
            public int Imprimirvida()
            {
                    /*se imprime el promedio de vida del deck completo y se
redondea al entero*/
                    int sumavida = 0;
                    for(int i = 1; i <= 8; i++)
                    {
                            int vida = int.Parse(deck[i-1, 0]);
                            sumavida += vida;
                    }
                    return (int)Math.Round((double)sumavida/8);
            }
            public int Imprimirdaño()
            {
          /*se imprime el promedio de daño del deck completo y se redondea al
entero*/
        int sumadaño = 0;
        for (int i = 1; i <= 8; i++)
        {
           int vida = int.Parse(deck[i-1, 0]);
           sumadaño += vida;
        }
        return (int)Math.Round((double)sumadaño/8);
    }

            public static string Obtenerganador(Decks deckjugador1, string
nickname1, Decks deckjugador2, string nickname2)
            {
```

```
                /*crear una condición para determinar el ganador dependiendo quien tiene mayor puntos de daño*/
                if (deckjugador1.Imprimirdaño() > deckjugador2.Imprimirdaño())
                {
                        return nickname1 + " Es el ganador";
                }

                else if (deckjugador1.Imprimirdaño() < deckjugador2.Imprimirdaño())
                {
                        return nickname2 + " Es el ganador";
                }
                else
                {
                        return "Empate";
                }
        }

        public static bool Decknulo(Decks decks)
        {
                /*sera falso cuando los decks no tengan ningun valor*/
                int filas = decks.deck.GetLength(0);
                int columnas = decks.deck.GetLength(1);
                for (int i = 1; i <= filas; i++)
                {
                        for (int j = 1; j <= columnas; j++)
                        {
                                if (decks.deck[i-1,j-1] == null)
                                {
                                        return false;
                                }
                        }
                }
                return true;
```

```
        }
    }
}
```

- Programa

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Proyecto_Lab_Progra;


namespace Proyecto_lab_Progra
{
    internal class Program
    {
        static void Main(string[] args)
        {
            string nombreEquipo1 = "";
            string nombreEquipo2 = "";
            string nickname1 = "";
            string nickname2 = "";
            int jugador1eleccion = 0;
            int jugador2eleccion = 0;
            string nickNameJ1 = "";
            string nickNameJ2 = "";

            Decks decks1 = new Decks();
            Decks decks2 = new Decks();
            Decks decks3 = new Decks();
            Decks decks4 = new Decks();
```

```
        while (true)
        {
            /*crear un menú*/
            Console.Clear();
            Console.WriteLine("Menú del juego");
            Console.WriteLine("1. Agregar jugadores");
            Console.WriteLine("2. Ver jugadores");
            Console.WriteLine("3. Agregar Decks");
            Console.WriteLine("4. Ver Decks");
            Console.WriteLine("5. Modificar Deck");
            Console.WriteLine("6. Escoger Mazo");
            Console.WriteLine("7. Jugar");
            Console.WriteLine("8. Terminar juego");
            Console.WriteLine("\n");
            Console.Write("Seleccione una opción: ");
            int opcion = int.Parse(Console.ReadLine());
            switch (opcion)
            {
                /*ingresar los nombres de los jugadores y los equipos para nombrar los
nicknames*/
                case 1:
                    Console.Clear();
                    Console.WriteLine("Ingrese el nickname del jugador 1");
                    nickname1 = Console.ReadLine();
                    Console.WriteLine("Ingrese el nombre el equipo 1");
                    nombreEquipo1 = Console.ReadLine();
                    string inicial1 = nombreEquipo1.Substring(0, 1);
                      string inicial2 = nombreEquipo1.Substring(nombreEquipo1.IndexOf('
') + 1, 1);
                    int length = nickname1.Length - 1;
                      string nicknamen = nickname1.Substring(nickname1.IndexOf(' ') + 1,
1);

                    Console.WriteLine("Ingrese el nickname del jugador 2");
```

```
                nickname2 = Console.ReadLine();
                Console.WriteLine("Ingrese el nombre el equipo 2");
                nombreEquipo2 = Console.ReadLine();
                string inicial12 = nombreEquipo2.Substring(0, 1);
                 string inicial22 = nombreEquipo2.Substring(nombreEquipo2.IndexOf('
') + 1, 1);
                int largo = nickname1.Length - 1;
                string nicknamen2 = nickname2.Substring(nickname1.IndexOf(' ') + 1,
1);

                    nickNameJ1 = inicial1 + inicial2 + " - " + nickname1 + length +
nicknamen + "0";
                   nickNameJ2 = inicial12 + inicial22 + " - " + nickname2 + largo +
nicknamen2 + "0";
                Console.WriteLine("\n");
                Console.WriteLine("Ya se han registrado los nombres y equipos");
                Console.WriteLine("\n");
                      Console.WriteLine("Presione cualquier tecla para regresar al
menu...");
                Console.ReadKey();
                break;
            case 2:
                /*visualizar los nicknames de los jugadores*/
                Console.Clear();
                if (nickNameJ1 != "" && nickNameJ1 != "")
                {
                    Console.WriteLine("Nombre del jugador 1: " + nickname1);
                    Console.WriteLine("Nombre del equipo 1: " + nombreEquipo1);
                    Console.WriteLine("Nickname del jugador 1: " + nickNameJ1);

                    Console.WriteLine("\nNombre del jugador 2: " + nickname2);
                    Console.WriteLine("Nombre del equipo 2: " + nombreEquipo2);
                    Console.WriteLine("Nickname del jugador 2: " + nickNameJ2);
                }
```

```
                    /*si el usuario ingresa al menu #2 antes de agregar los jugadores, le saldra un aviso que debe ingresar la información antes*/
                    else
                    {
                        Console.WriteLine("Debe agregar jugadores");
                    }

                    Console.WriteLine("\n");
                            Console.WriteLine("Presione cualquier tecla para regresar al menu...");
                    Console.ReadKey();
                    break;
                case 3:
                    /*ingresar unicamente la información de los cuatro decks*/
                    Console.Clear();
                    Console.WriteLine("Deck #1");
                    for (int i = 1; i <= 8; i++)
                    {
                        string nom; string tipo; int elixir; int pvida; int daño;

                        Console.WriteLine("Ingrese el nombre de la carta " + i);
                        nom = Console.ReadLine();
                        Console.WriteLine("Ingrese el tipo de carta " + i);
                        tipo = Console.ReadLine();
                        Console.WriteLine("Cuantos puntos de vida tiene " + i);
                        pvida = Convert.ToInt32(Console.ReadLine());
                        Console.WriteLine("Ingrese el elixir perteneciente a la carta " + i);
                        elixir = Convert.ToInt32(Console.ReadLine());
                        Console.WriteLine("Ingrese los puntos de daño que genera la carta " + i);
                        daño = Convert.ToInt32(Console.ReadLine());
                        decks1.AddCard(pvida, elixir, daño, nom, tipo, i - 1);
                    }
                    Console.WriteLine("\n");
```

```
                Console.WriteLine("************************************************");
                Console.WriteLine("\nDeck #2");
                for (int i = 1; i <= 8; i++)
                {
                    string nom; string tipo; int elixir; int pvida; int daño;

                    Console.WriteLine("Ingrese el nombre de la carta " + i);
                    nom = Console.ReadLine();
                    Console.WriteLine("Ingrese el tipo de carta " + i);
                    tipo = Console.ReadLine();
                    Console.WriteLine("Cuantos puntos de vida tiene " + i);
                    pvida = Convert.ToInt32(Console.ReadLine());
                    Console.WriteLine("Ingrese el elixir perteneciente a la carta " + i);
                    elixir = Convert.ToInt32(Console.ReadLine());
                    Console.WriteLine("Ingrese los puntos de daño que genera la carta
" + i);
                    daño = Convert.ToInt32(Console.ReadLine());
                    decks2.AddCard(pvida, elixir, daño, nom, tipo, i - 1);
                }
                Console.WriteLine("\n");
                Console.WriteLine("************************************************");

                Console.WriteLine("\nDeck #3");
                for (int i = 1; i <= 8; i++)
                {
                    string nom; string tipo; int elixir; int pvida; int daño;

                    Console.WriteLine("Ingrese el nombre de la carta " + i);
                    nom = Console.ReadLine();
                    Console.WriteLine("Ingrese el tipo de carta " + i);
                    tipo = Console.ReadLine();
                    Console.WriteLine("Cuantos puntos de vida tiene " + i);
                    pvida = Convert.ToInt32(Console.ReadLine());
                    Console.WriteLine("Ingrese el elixir perteneciente a la carta " + i);
```

```
                elixir = Convert.ToInt32(Console.ReadLine());
                Console.WriteLine("Ingrese los puntos de daño que genera la carta
" + i);
                daño = Convert.ToInt32(Console.ReadLine());
                decks3.AddCard(pvida, elixir, daño, nom, tipo, i - 1);
            }
            Console.WriteLine("\n");
            Console.WriteLine("************************************************");

            Console.WriteLine("\nDeck #4");
            for (int i = 1; i <= 8; i++)
            {
                string nom; string tipo; int elixir; int pvida; int daño;

                Console.WriteLine("Ingrese el nombre de la carta " + i);
                nom = Console.ReadLine();
                Console.WriteLine("Ingrese el tipo de carta " + i);
                tipo = Console.ReadLine();
                Console.WriteLine("Cuantos puntos de vida tiene " + i);
                pvida = Convert.ToInt32(Console.ReadLine());
                Console.WriteLine("Ingrese el elixir perteneciente a la carta " + i);
                elixir = Convert.ToInt32(Console.ReadLine());
                Console.WriteLine("Ingrese los puntos de daño que genera la carta
" + i);
                daño = Convert.ToInt32(Console.ReadLine());
                decks4.AddCard(pvida, elixir, daño, nom, tipo, i - 1);
            }
            Console.WriteLine("\n");
            Console.WriteLine("Agregaste todos los decks");
            Console.WriteLine("\n");
                    Console.WriteLine("Presione cualquier tecla para regresar al
menu...");
            Console.ReadKey();
            Console.Clear();
```

```
                    break;
                case 4:
                    /*Visualizar el promedio total de puntos de daño y puntos de vida de cada deck*/
                    Console.Clear();
                        if (Decks.Decknulo(decks1) && Decks.Decknulo(decks2) && Decks.Decknulo(decks3) && Decks.Decknulo(decks4))
                    {
                        Console.WriteLine("Deck #1");
                                Console.WriteLine("El promedio de daño es: " + decks1.Imprimirdaño());
                                 Console.WriteLine("El promedio de vida es: " + decks1.Imprimirvida());

                        Console.WriteLine("\nDeck #2");
                                Console.WriteLine("El promedio de daño es: " + decks2.Imprimirdaño());
                                 Console.WriteLine("El promedio de vida es: " + decks2.Imprimirvida());

                        Console.WriteLine("\nDeck #3");
                                Console.WriteLine("El promedio de daño es: " + decks3.Imprimirdaño());
                                 Console.WriteLine("El promedio de vida es: " + decks3.Imprimirvida());

                        Console.WriteLine("\nDeck #4");
                                Console.WriteLine("El promedio de daño es: " + decks4.Imprimirdaño());
                                 Console.WriteLine("El promedio de vida es: " + decks4.Imprimirvida());
                    }
                    else
                    {
```

```
                    Console.WriteLine("Debe agregar los decks");
                }

                Console.WriteLine("\n");
                        Console.WriteLine("Presione cualquier tecla para regresar al
menu...");
                Console.ReadKey();
                break;
            case 5:
                /*Modificar la carta deseada de un deck específico*/
                Console.Clear();
                        if (Decks.Decknulo(decks1) && Decks.Decknulo(decks2) &&
Decks.Decknulo(decks3) && Decks.Decknulo(decks4))
                {
                    Console.Write("Ingrese el número de deck que quiere modificar ");
                    int deckmodificar = int.Parse(Console.ReadLine());
                    Console.Write("Ingrese el número de carta que quiere modificar ");
                    int cartamodificar = int.Parse(Console.ReadLine());

                    string nombremodificado; string tipomodificado; int elixirmodificado;
int pvidamodificado; int dañomodificado;
                    Console.WriteLine("\nIngrese el nombre de la carta ");
                    nombremodificado = Console.ReadLine();
                    Console.WriteLine("Ingrese el tipo de carta ");
                    tipomodificado = Console.ReadLine();
                    Console.WriteLine("Cuantos puntos de vida tiene ");
                    pvidamodificado = Convert.ToInt32(Console.ReadLine());
                    Console.WriteLine("Ingrese el elixir perteneciente a la carta ");
                    elixirmodificado = Convert.ToInt32(Console.ReadLine());
                    Console.WriteLine("Ingrese los puntos de daño que genera la carta
");
                    dañomodificado = Convert.ToInt32(Console.ReadLine());
```

```
                    /*Con el deck elegido, buscar el dato de la carta dependiendo del
numero del deck y carta*/
                    if (deckmodificar == 1)
                    {
                        decks1.deck[cartamodificar - 1, 0] = pvidamodificado.ToString();
                        decks1.deck[cartamodificar - 1, 1] = dañomodificado.ToString();
                        decks1.deck[cartamodificar - 1, 2] = elixirmodificado.ToString();
                                                    decks1.deck[cartamodificar - 1, 3] =
nombremodificado.ToString();
                        decks1.deck[cartamodificar - 1, 4] = tipomodificado.ToString();
                    }
                    else if (deckmodificar == 2)
                    {
                        decks2.deck[cartamodificar - 1, 0] = pvidamodificado.ToString();
                        decks2.deck[cartamodificar - 1, 1] = dañomodificado.ToString();
                        decks2.deck[cartamodificar - 1, 2] = elixirmodificado.ToString();
                                                    decks2.deck[cartamodificar - 1, 3] =
nombremodificado.ToString();
                        decks2.deck[cartamodificar - 1, 4] = tipomodificado.ToString();
                    }
                    else if (deckmodificar == 3)
                    {
                        decks3.deck[cartamodificar - 1, 0] = pvidamodificado.ToString();
                        decks3.deck[cartamodificar - 1, 1] = dañomodificado.ToString();
                        decks3.deck[cartamodificar - 1, 2] = elixirmodificado.ToString();
                                                    decks3.deck[cartamodificar - 1, 3] =
nombremodificado.ToString();
                        decks3.deck[cartamodificar - 1, 4] = tipomodificado.ToString();
                    }
                    else if (deckmodificar == 4)
                    {
                        decks4.deck[cartamodificar - 1, 0] = pvidamodificado.ToString();
                        decks4.deck[cartamodificar - 1, 1] = dañomodificado.ToString();
                        decks4.deck[cartamodificar - 1, 2] = elixirmodificado.ToString();
```

```
                                                    decks4.deck[cartamodificar - 1, 3] =
nombremodificado.ToString();
                        decks4.deck[cartamodificar - 1, 4] = tipomodificado.ToString();
                    }
                            Console.WriteLine("El deck #" + deckmodificar + " ha sido
modificado");
                }
                else
                {
                    Console.WriteLine("Debe agregar el deck a modificar");
                }

                Console.WriteLine("\n");
                        Console.WriteLine("Presione cualquier tecla para regresar al
menu...");
                Console.ReadKey();
                break;
            case 6:
                        /*Escoger el mazo de cada jugador con el numero de deck
dependiendo en el orden en que lo agrego al sistema*/
                Console.Clear();
                if (nickNameJ1 != "" && nickNameJ2 != "")
                {
                    Console.Write("Mazo jugador 1: ");
                    jugador1eleccion = int.Parse(Console.ReadLine());
                    Console.Write("\nMazo jugador 2: ");
                    jugador2eleccion = int.Parse(Console.ReadLine());
                }
                else
                {
                    Console.WriteLine("Debe agregar jugadores");
                }
                Console.WriteLine("\n");
```

```
                    Console.WriteLine("Presione cualquier tecla para regresar al
menu...");
                Console.ReadKey();
                break;
            case 7:
                    /*en esta sección del menú se evalúa que jugador ganará
dependiendo del deck escogido*/
                Console.Clear();
                if ((jugador1eleccion > 0) && (jugador2eleccion > 0))
                {
                    Decks deckjugador1 = new Decks();
                    Decks deckjugador2 = new Decks();
                    switch (jugador1eleccion)
                    {
                        case 1:
                            deckjugador1 = decks1;
                            break;
                        case 2:
                            deckjugador1 = decks2;
                            break;
                        case 3:
                            deckjugador1 = decks3;
                            break;
                        case 4:
                            deckjugador1 = decks4;
                            break;
                    }
                    switch (jugador2eleccion)
                    {
                        case 1:
                            deckjugador2 = decks1;
                            break;
                        case 2:
                            deckjugador2 = decks2;
```

```
                        break;
                    case 3:
                        deckjugador2 = decks3;
                        break;
                    case 4:
                        deckjugador2 = decks4;
                        break;
                }
                        string ganador = Decks.Obtenerganador(deckjugador1,
nickname1, deckjugador2, nickname2);
                    Console.WriteLine(ganador);
                }
                else
                {
                    Console.WriteLine("Debe seleccionar el deck para cada jugador");
                }
                Console.WriteLine("\n");
                        Console.WriteLine("Presione cualquier tecla para regresar al
menu...");
                Console.ReadKey();
                break;
            case 8:
                /*en esta sección le permite al jugador terminar el juego*/
                Environment.Exit(0);
                break;
        }
    }
  }
}
```

VI. EXPLICACION DEL CODIGO

- Clase Deck

En la clase denominada “decks” se declararon todas las variables privadas a excepcion de una: deck la cual es una matriz para utilizar en el programa principal. Luego se declaró el constructor como matriz para guardar la información de cada carta de cada deck.

Después se agregó un método llamado “addcard” para que el programa guarde la información específica de todas las cartas. Con los parametros de puntos de vida, elixir, daño, nombre y tipode la carta y posicion, la ultima es para especificar el lugar donde se almacenará dicha información.

Los siguientes metodos que se crearon son para imprimir los puntos de daño y de vida donde se colocó un bucle “for” donde se suman todos los datos y se dividen en 8 el cual es el total de cartas para cada mazo, también se agregó la función para redondear al entero más cercano para que sea mas sencillo calcular el ganador.

Posteriormente se creo el método de obtener ganador, en donde se creo una condición (if-else) para determinar el ganador dependiendo quien tiene mayor puntos de daño y si ambos tienen la misma cantidad de puntos de daño, entonces imprimirá un empate.

Por último se diseño el método deck nulo como bool para verificar si todos los elementos de la matriz bidimensional son distintos de nulo o null, donde será falso cuando los deck no tengan ningún valor y verdadero cuando si lo tengan.

- Programa

En el programa principal se programó un menú principal para tener más ordenado cada información que se solicita. Se creo un modo switch para cada caso u opcion del menú. De primero esta la opción para agregar a los jugadores y sus equipos, para que de este modo se creen los nicknames. Utilizando funciones substring e indexof para las iniciales de cada equipo y el largo de cada nombre.

La segunda opción del menú es ver los jugadores donde se visualizan el nombre de los jugadores, los equipos y los nicknames. Si el usuario ingresa al menu #2 antes de agregar los jugadores, le saldra un aviso que debe ingresar la información antes.

El tercer caso del switch es unicamente agregar la información de los 4 decks: el nombre, tipo, puntos de vida, puntos de daño y elixir de la carta.

La cuarta opción es para ver los decks, donde se visualiza el promedio total de puntos de daño y puntos de vida de cada deck utilizando la opción “decks1.Imprimirdaño” para invocar al metodo de la clase anteriormente creado. Si el usuario ingresa al menu #4 antes de agregar los decks, le saldra un aviso que debe ingresar la información antes.

El quinto caso es para modificar la carta deseada de un deck específico donde primero se le pregunta al jugador que carta de que deck quiere modificar para mejorar su mazo, luego que ingrese toda la información de la carta nueva y con el deck elegido, buscar el dato de la carta dependiendo del numero del deck y carta.

El caso numero 6 se creó una condición que si ya se ingresaron los jugadores es para que los estos jugadores puedan escoger su deck con el numero de deck dependiendo en el orden en que lo agrego al sistema. y si no, saldra un mensaje que debe ingresar dichos jugadores.

El septico case es para identificar quien es el ganador. en esta sección del menú se evalúa que jugador ganará dependiendo del deck escogido y los puntos de daño que tiene cada uno. Donde se crearon dos switch adentro para evaluar cada opción que existe con cada jugador y determinar el ganador.

Finalmente en el ultimo case del switch existe la opcion para salir del juego y terminar de jugar.