Programacion Funcional
IIC 3585-1

Grupo 4

- Vicente Chadwick, Felipe Trejo, Lucas Rodríguez

Funcionamiento General

Tres funciones principales que encapsulan el funcionamiento del juego

```
function init_game() {
  // inicia el juego
  return new_players;
}

function new_players() {
  // Con una funcion recursiva pregunta por el numero
  // de jugadores y sus nombres
  return play_game;
}

function play_game() {
  // Con una funcion recursiva recibe las jugadas hasta
  // que algun jugador llegue a 0 puntos
  return 'winner';
}

init_game()()
```

Programación funcional implementada

- Funciones de primer orden
- Funciones de orden superior
- Curring
- Recursividad
- Closure

Librerias utilizadas

- readline-sync
- json-parse
- lodash/fp/conmpose

```
//Importamos librerias
var reader = require('readline-sync');
var parse = require('json-parse');
var compose = require('lodash/fp/compose');
```

Funciones de primera clase

La mayoría de las funciones del programa son de primera clase.

```
// Tiros especiales
const specials = {
    "DB": 50,
    "SB": 25
}

//Obtiene nombre de jugador
const get_name_player =  () => reader.question(`Cual es el nombre del jugador? `);

//Retorna true si tiro es especial, false en caso contrario
const check_specials = shot => shot in specials? true : false;

//Calcula puntos de cierto tiro
const get_points = shot => check_specials(shot)? specials[shot]: shot[0] * shot[1];

//Obtiene jugada
const get_play = name => reader.question(`Jugador ${name} es tu turno. Ingresa tu jugada `);

//Funcion que crea un jugador
const init_player = () => {
    const name = get_name_player()
    const player = {
```

Función de orden superior

```
 curring -> Dado un argumento "shots" (definido previamente) retorna una función con argumento "p
iza para verificar si el jugador ha ganado debido a una jugada especifica.
rt_play = (player) => {
sult = 0;
parsed = parse.bind(console, []);
compute_values = (shots) => {
 Uso de Compose y funcion de Orden superior (forEach)
mpose((shots_array) => shots_array.forEach(shot => result +=  get_points(shot)), parsed)(shots)
ayer.points = Math.abs(player.points - result)
nsole.log(`Ahora tienes ${player.points} puntos`)
turn player.points === 0

 compute_values
```

Recursividad y Currying

- Con la **recursividad** y la operación condicional (ternaria) podemos simular un while en una sola línea.
- Podemos observar que en primer lugar a insert_play le otorgamos un jugador. Esta devuelve una función a la cual se le pasan los tiros para que calcule el puntaje del jugador entregado anteriormente

```
// Simula una jugada para cada jugador. Retorna si existe ganador o no.
const play_game = (players) => {
    let winner_found = null
    const is_there_winner = () => {
        // Funcion de orden superior forEach
        players.forEach((player) => {
            const insert_play_of_player = insert_play(player)
            // Funcion que evalua si existe ganador. Retorna el ganador si es que hay sino null.
            // Utiliza Funcion Compose y Recursividad para busqueda de ganador (flujo juego)
            winner_found = compose((shots) => insert_play_of_player(shots), get_play)(player.name) == true ? player : winner_found
        })
        return winner_found ? true : false
    }

    // Recursividad para ejecución juego. Retorna mensaje cuando gana jugador.
    const find_winner = () => !is_there_winner() ? find_winner() : undefined;
    if (players.length !== 0) {
        find_winner()
        return console.log(`Felicidades ${winner_found.name} haz ganado esta partida`)
    }
    console.log("No habían jugadores para esta partida")
}
```

Closure y Compose

- Gracias a los beneficios que nos da ***closure***, es que podemos setear en un principio los puntajes y nombres de los usuarios sin posibilidad que se cambien dentro del juego.
- Por otro lado, ocupamos ***compose*** en varias partes del código, lo que hace más fácil la lectura del código y explicitar su flujo.

```
// Da bienvenida al juego y se agregan jugadores. Una vez finalizada se ejecuta función play_game
const init_game = () => {
    console.log('Bienvenidos al juego\n')
    const players = []
    const new_player = () => reader.question('Desea agregar un jugador (1 Si; 0 No): ') === "1"
    // Funcion Compose y Recursividad para agregar jugadores
    const new_players = () => new_player() ?  (compose((player) => players.push(player),init_player)(), new_players()): () => play_game(players)
    return new_players
```

Programacion Funcional
IIC 3585-1

Grupo 4

- Vicente Chadwick, Felipe Trejo, Lucas Rodríguez