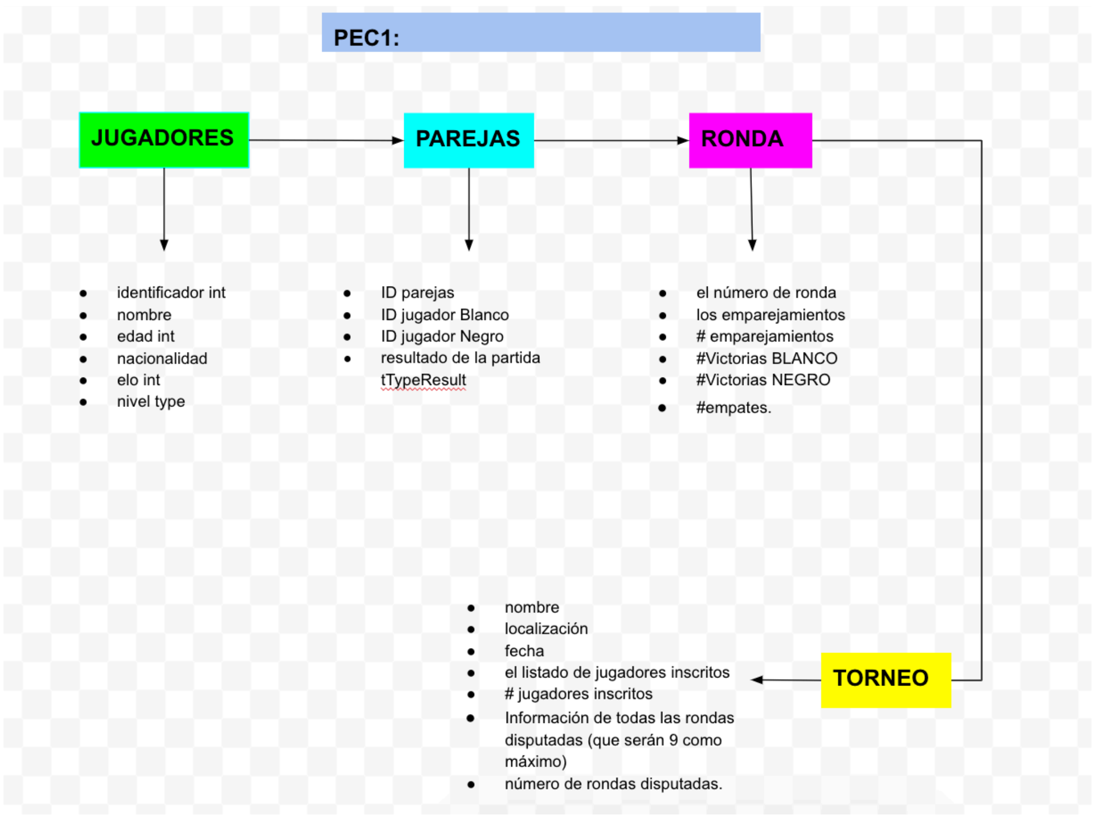

**PEC1**

## [20%] Ejercicio 1: Definición de tipo de datos

```
const
        MAX_PAR : integer := 5;
        MAX_ROUND : integer := 9;
end const

type
  tTypeLevel = {novice, medium, advanced}
  tPlayer = record
        idPlayer: integer;
        namePlayer: string;
        age: integer;
        nationality: string;
        elo: integer;
        level: tTypeLevel;
  end record
  tTypeResult = {whiteWins, blackWins, draw}
  tPairing = record
        idPairing: integer;
        idWhite: integer;
        idBlack: integer;
        result: tTypeResult;
   end record



  tRound = record
        roundId: integer;
        paring: vector [MAX_PAR] of tPairing;          //máximo 5 emparejamientos//
        pairingNum: integer;
        whiteWinsNum: integer;
        blackWinsNum: integer;
        drawWinsNum: integer;
   end record
end type



tDate = record
        day : integer;
        month : integer;
        year : integer;
end record


  tChessTournament = record
        chessTournamentName: string;
        location: string;
        date: tDate;
        playerList: pointer to tPlayer;
        playersNum: integer;
        rounds: vector [MAX_ROUND] of tRound;          //max 9 rondas disputadas//
        roundsNum: integer;
   end record
end type
```

## Ejercicio 3: Especificación formal

**A)**

**action** init_chess_tournamen(**in/out** c : tChessTournament)

PRE : {c = C}

*No hay que definir variables pues todas vienen como parámetros*

POST: {c.playersNum = 0 y c.roundsNum  = 0 }

**B)**

**action** new_player (**in/out** c: tChessTournament ; **in** idPlayer: **integer**; **in** name: **string**; **in** age: **integer; in** nationality: **string; in** elo: **integer**)

Pre:{c = C y idPlayer = IDPLAYER y ID>0 y name = NAME y age = AGE y nationality = NATIONALITY y elo=ELO}

Post: { (∃i : 0 < i ≤ C.numPlayer : c.numPlayer = c.numPlayer y c.playerList[i].idPlayer = ID) o (c.numPlayer= C.numPlayer+1 y c.playerList[c.numPlayer].idPlayer = IDPLAYER y c.playerList[c.numPlayer].name = NAME y c.playerList[c.numPlayer].age = AGE y c.playerList[c.numPlayer].nationality = NATIONALITY y c.playerList[c.numPlayer].elo = ELO)}

*Asserts añadidos a la solución CodeLite adjunta

**Ejercicio 4: Diseño descendente**

**Nivel 1:**

```
action levels_winners (c:tChessTournament)

var
   i: integer;
   j: integer;
end var

j:= 1;

for i:= 1 to c.roundsNum do

        while j ≤ c.rounds[j].pairingNum AND  c.rounds[j].paring[j].result ≠ draw do

                if find_player(c, idPlayer) = idWhite then

                        show_levels_winners (c, idWhite);

                end if

                if find_player(c, idPlayer) = idBlack then

                show_levels_winners (c, idBlack);

                end if

         j:= j+1;

         end while

    end for
end action
```

**Nivel 2:**

```
action find_player (c: tChessTournament, idPlayer: integer)

var
   player: pointer to tPlayer;
   i: integer;
   j:integer;
end var

i: = 1;
player: = NULL;

while i < c.playersNum AND player = NULL do
            if c.playerList[i].idPlayer = idPlayer then
            player = c.playerList[i] ;
            else
            i:= i +1;
   end if
end while
```

**Nivel 3:**

```
action show_levels_winners (idPlayer: integer)
            writeString ("ID PLAYER: ");
            writeInteger (idPlayer);
            writeString ("PLAYER NAME: ");
            writeInteger (namePlayer);
            writeString ("PLAYER LEVEL: ");
            writeInteger (level);
end action
```