Tema 6. Juegos

Ejemplos

1. Juego de las monedas que suman una cantidad
2. Juego del tablero con casillas para moverse y obstáculos que lo impiden

Monedas (ejercicio 5)

Enunciado

- Sobre la mesa tenemos un montón de ***N*** monedas, cada una de un determinado valor. Dos jugadores cogen alternativamente una moneda del montón. Gana el primer jugador cuyas monedas suman exactamente una cantidad dada ***K***, prefijada de antemano.
- Construya un algoritmo que dada una configuración del juego la clasifique (*ganadora*, *perdedora* o *tablas*) y devuelva la decisión que produce el resultado indicado.
- NOTA
  - Puede haber varias monedas con el mismo valor.

Monedas (ejercicio 5)

Ejemplo

- Si el Montón inicial de monedas es (representadas por su valor) C = {1,2,2,3,4,5,6,6,7} y K = 15, el desarrollo del juego puede ser como sigue:
  - **A** elige 5 y queda **C** = {*1,2,2,3,4,6,6,7*} $\mathbf{S_a}$= 5 $\mathbf{S_b}$= 0
  - **B** elige 7 y queda **C** = {*1,2,2,3,4,6,6*} $\mathbf{S_a}$= 5 $\mathbf{S_b}$= 7
  - **A** elige 6 y queda **C** = {*1,2,2,3,4,6*} $\mathbf{S_a}$= 11 $\mathbf{S_b}$= 7
  - **B** elige 6 y queda **C** = {*1,2,2,3,4*} $\mathbf{S_a}$= 11 $\mathbf{S_b}$= 13
  - **A** elige 4 y ***Gana***, puesto que la suma de las monedas que ha elegido es $\mathbf{S_a}$=15 = (*5*+*6*+*4*)

Monedas (ejercicio 5)


```
void Evaluar(int Suma, int SumaA, int SumaB, int Monedas[], int N, TPosicion *Pos, int *Moneda){
        int i, x, Valor;                TPosicion PosDejamos;
        if ( (MontonVacio(Monedas, N))   ||  ((SumaA > Suma)  &&  (SumaB > Suma))  )  *Pos = Tablas;
        else {
            i = 0;          *Pos = Perdedora;
            while ( (i<N)  &&  (*Pos != Ganadora)) {
               if ( Monedas[i]  !=  0 ) {
                   if (  Suma   ==   SumaA + Monedas[i]) {   // La pos. es GANADORA: elegimos la moneda i
                          *Pos = Ganadora;        *Moneda = i;
                   } else {              // Hay que evaluar la jugada (Tomamos la moneda i)
                          Valor  = Monedas[i];                    Monedas[i] = 0;
                          Evaluar(Suma, SumaB, SumaA+Valor, Monedas, N, &PosDejamos, &x);
                          if ( PosDejamos != Ganadora ) {
                              *Moneda = i;
                              if ( PosDejamos == Perdedora) *Pos = Ganadora;
                              else *Pos = Tablas;
                          }
                          Monedas[i] = Valor;
                   }
               }
               i++;
            }
        }
}
```

```
Typedef  enum  {
    Perdedora,
    Tablas,
    Ganadora
}  TPosicion
```

Tablero (ejercicio 7)

Enunciado

- Tenemos un tablero de $M \times N$ casillas, en alguna de las cuales hay obstáculos.
- Dos jugadores juegan alternativamente al siguiente juego. Un jugador situado en una casilla puede desplazarse a cualquiera de las casillas adyacentes, tanto en diagonal, horizontal como verticalmente, que no tenga obstáculos y no haya pasado por ella anteriormente ninguno de los dos jugadores. Si un jugador no puede moverse pasa el turno.
- Gana el jugador que visita más casillas.
- Construya un algoritmo que evalúe el juego, para una posición del mismo dada, indicando la jugada a realizar.

Tablero (ejercicio 7)


```
void Evaluar (TJugador JugadorA, TJugador JugadorB, int Tab[MAXDIM][MAXDIM],
              int N, TPosicion *Pos, TCasilla *Cas) {
    int i, j;
    TPosicion PosDejamos;
    TCasilla NCas, CasX;
    TJugador Jugador;
    ……
```

```
typedef enum {
    Perdedora, Tablas, Ganadora
} TPosicion;

typedef struct {
    int Fila, Columna;
} TCasilla;

typedef struct {
    TCasilla Casilla;
    int NumCasillas;
} TJugador;
```

```
int Tablero [MAXDIM][MAXDIM] =
{
        1,1,0,1,1,1,1,1,1,1,
        1,1,0,1,1,1,0,0,1,1,
        1,1,0,0,1,1,1,1,1,1,
        1,1,1,0,1,1,0,1,1,1,
        1,1,1,1,1,0,1,1,1,1,
        0,0,0,1,1,0,1,1,1,1,
        0,0,0,1,1,1,0,0,1,1,
        0,0,0,1,1,1,0,1,1,1,
        0,0,0,1,1,1,0,1,1,1,
        1,1,1,1,1,1,0,1,1,1
};
```

Tablero (ejercicio 7)


```
void Evaluar (TJugador JugadorA, TJugador JugadorB, int Tab[MAXDIM][MAXDIM],
              int N, TPosicion *Pos, TCasilla *Cas) {
  .........
    if (Bloqueado(Tab, N, JugadorA.Casilla)) // JugadorA no puede moverse
        if (JugadorA.NumCasillas < JugadorB.NumCasillas)
          *Pos = Perdedora;
        elseif (Bloqueado(Tab, N, JugadorB.Casilla)) // JugadorB no puede moverse
            if (JugadorA.NumCasillas == JugadorB.NumCasillas)
              *Pos = Tablas;
            else *Pos = Ganadora;
        else { // JugadorB puede moverse, luego jugará
            Evaluar(JugadorB, JugadorA, Tab, N, &PosDejamos, &CasX);
            if (PosDejamos == Ganadora)  *Pos = Perdedora;
            else if (PosDejamos == Perdedora) *Pos = Ganadora;
            else *Pos = Tablas;
          }
     else {
    .........
```

Tablero (ejercicio 7)


```
void Evaluar (TJugador JugadorA, TJugador JugadorB, int Tab[MAXDIM][MAXDIM],
              int N, TPosicion *Pos, TCasilla *Cas) {

  .........
                i = -1;                  *Pos = Perdedora;
                while ( (i <= 1)  &&  (*Pos != Ganadora) )  {
                  j = -1;
                  while ( (j <= 1)  &&  (*Pos != Ganadora) )  {
                      NCas = NuevaCasilla(JugadorA.Casilla.Fila+i , JugadorA.Casilla.Columna+j);
                      if ( Valida(Tab, N, NCas) ) {
                         Tab[NCas.Fila][NCas.Columna] = FALSE;
                         Jugador = JugadorA;          JugadorA.Casilla = NCas;     JugadorA.NumCasillas++;
                         Evaluar(JugadorB,JugadorA,Tab, N, &PosDejamos, &CasX);
                         if (PosDejamos != Ganadora) {
                              *Cas = NCas;
                              if (PosDejamos == Perdedora) *Pos = Ganadora;
                              else *Pos = Tablas;
                          }
                         Tab[NCas.Fila][NCas.Columna] = TRUE;          JugadorA = Jugador;
                       }
                       j++;
                    }
                    i++;
                  }   }   } // Fin Evaluar
```