Informe de Ejemplo
Juego Nim

75.40 Algoritmos y Programación I
Cátedra Wachenchauzer

Cuatrimestre I, 2014

1. Dificultades a afrontar

Luego de analizar el enunciado del trabajo práctico, se encontraron las siguientes dificultades, las cuales serán necesario analizar cómo resolver, teniendo en cuenta que la resolución de una podría limitar la resolución de otra:

1. La forma de representar los datos del tablero, de manera lógica.
2. La metodología para mostrar el tablero (representación gráfica) de la manera pedida por el enunciado.
3. La *inteligencia artificial* del *jugador computadora*.

2. Resolución

La primer forma que salta a la mente a la hora de representar los datos es la trivial: una matriz (o lista de listas). Esto tiende a ser bastante simple: un usuario dice que quiere N pilas, entonces se crean N listas, y por cada una se va pidiendo la cantidad de fichas, agregando esa cantidad de *fichas* (sea como fuera que las vayamos a representar, por ejemplo, 'O'). Cuando un usuario nos pide de sacar una cantidad k de fichas a una determinada pila, simplemente le sacamos esa cantidad de elementos a la lista correspondiente.

Este método tiene la ventaja de ser una solución bastante simple. La desventaja que tiene es la cantidad de espacio de memoria utilizado: teniendo N pilas, suponiendo que cada pila tiene M fichas, se está consumiendo lugar para N*M valores (del tipo de dato a guardar correspondiente). Ésto es claramente ineficiente, puesto que no nos interesa tener las fichas en sí (en la parte lógica del programa), sino **cuántas** fichas tiene cada pila. Y he allí de donde nace una segunda implementación posible: en vez de tener una matriz, utilizamos una lista, que tenga en orden la cantidad de fichas que le quedan a una determinada pila. Cuando el usuario selecciona una pila para sacarle k fichas, simplemente restamos la cantidad k al actual (previas validaciones correspondientes). Por lo que se puede observar, la forma de tratar lógicamente al tablero es bastante simple. La desventaja con el método anterior, posiblemente, radicaría en que la representación en forma de matriz tiende a ser más intuitiva, por ser más parecida a la realidad, pero vemos que almacenar las fichas en sí no nos agrega información.

Analizamos en cada implementación, cómo deberíamos actuar para representarlo de la manera gráfica correspondiente:

1. Con el método de matriz: Tendríamos que conseguir el valor máximo de longitud/altura de una de las pilas. Luego ir verificando para cada una de las pilas si tienen esa altura o mayor (caso positivo, dibuja 'O', caso negativo, un espacio en blanco). Luego se disminuye en 1 la altura y se prosigue nuevamente, para todas las pilas, hasta llegar a 1 (caso mínimo)[1].

[1]Otra forma de realizar la representación gráfica con el método de la matriz es teniendo directamente una matriz de cadenas, con 'O' donde haya una ficha y espacios en blanco para rellenar en altura (hasta la máxima altura). Esto permite simplemente iterar la matriz para imprimirla, pero limita mucho por el lado de la representación lógica (sin contar que será necesario tener otra lista para recordar la cantidad de fichas que tiene la pila, para poder realizar las validaciones correspondientes). Por estas razones, esta implementación queda descartada.

2. Con el método de lista de cantidades: La idea es esencialmente la misma, pero nos quedaremos con la máxima cantidad, pues representa directamente la altura, luego vamos analizando pila por pila (o elemento por elemento de la lista), de la misma manera que se hace en el caso anterior.

Como se puede ver, implementar el tablero con una lista de cantidades/alturas no implica una mayor complicación en el momento de la representación gráfica (lo cual también es importante). Por todo esto, y por ser claramente una mejor opción (además de brindar más flexibilidad lógica al modelo), se elige la segunda opción de representación.

Para la inteligencia artificial del modo computadora, se pueden optar por muchas ideas, pero trataremos de ser simples con esto, puesto que si pensara en implementar una solución de mayor compromiso, la finalización a tiempo del trabajo práctico puede verse en riesgo innecesariamente (puesto que no se pedía nada demasiado elaborado). Por lo tanto se piensa en la siguiente idea: Se elige la pila de mayor cantidad de fichas (si hay más de una con esa cantidad, se elije la primera), y de esa pila se sacan todas las fichas, exceptuando una, a menos que sólo quedara una ficha en esa pila, o sólo quedara una pila con fichas (en cuyo caso se sacan todas las fichas). La idea es bastante simple, pero vislumbra un tipo de estrategia válida (sin recurrir simplemente a sacar una cantidad de fichas aleatoria de una pila aleatoria). Obviamente puede tener muchas mejoras agregando algo de aleatoriedad, pero con esta inteligencia se logra lo pedido por el enunciado.

3. Flujo del programa principal

En la figura 1 mostramos un diagrama representativo del flujo del programa principal. Por cuestiones de simplicidad y mejor visualización, no se agregan detalles como pedidos de datos a los usuarios, ni sus validaciones (y que en caso de ser erróneas, vuelven al paso anterior).

Aclaraciones:

- Sobre *"Validación de entrada"*: Es importante notar que cada una de las entradas será validada correspondientemente. Además de validar que se traten de valores numéricos, se pondrán valores máximos para que el juego mantenga coherencia, y evitar que nuestra simple representación gráfica no tenga mayores problemas (tener una cantidad inmensa de pilas, puede complicar por el lado horizontal, y dejar que una pila tenga 500 fichas puede impedir la visualización por el lado vertical).
- Sobre *"Juega a quien le toca"*: En caso que se trate de modo 2 jugadores, en ambos casos se le pide (y validan) los datos a los usuarios. En modo 1 jugador, cuando le toque al usuario se le pedirá a éste que ingrese los datos (los cuales se validarán) y en caso que le toque a 'Dr. Memory', realizará su movimiento como ya fue explicado anteriormente.

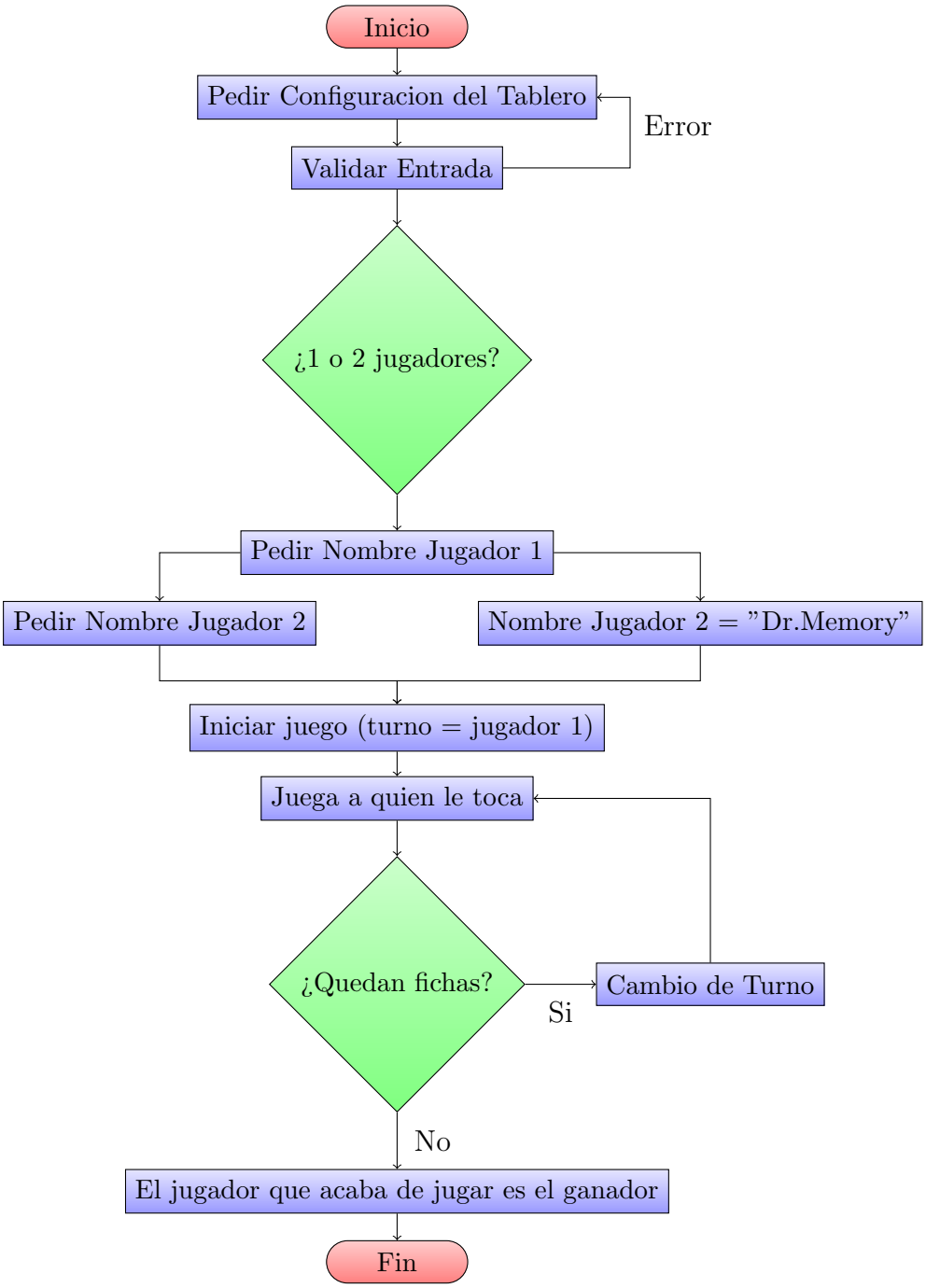

Figura 1: Diagrama de Flujo