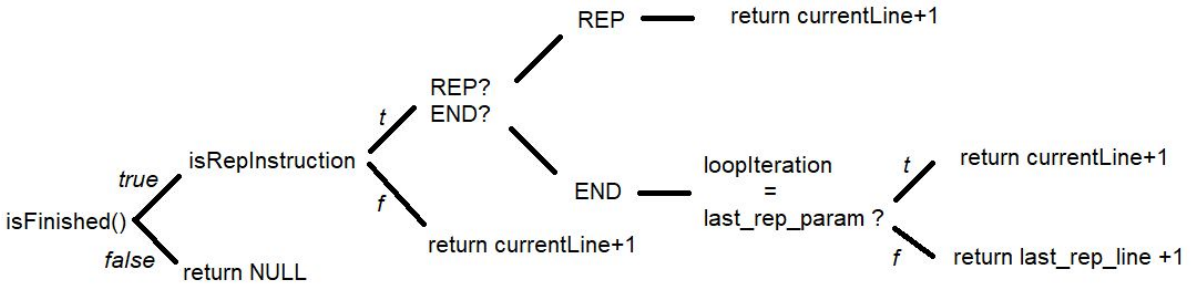
LAB II POO

Dificultad principal

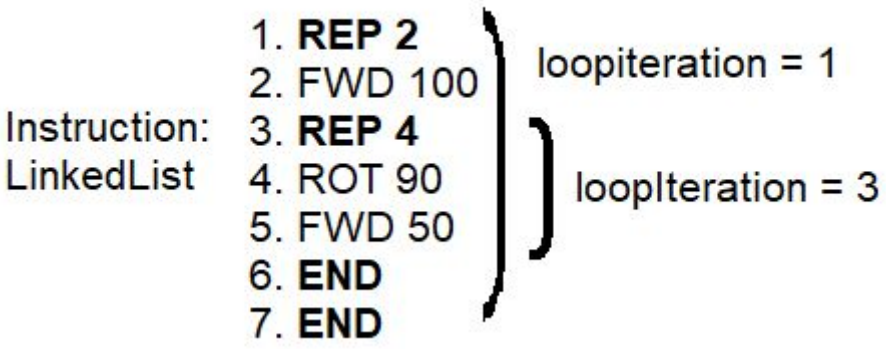
Nuestro mayor dificultad al desarrollar el código la hemos encontrado en el metodo **getNextInstruction()** de la clase Program.

Al ejecutar este método el programa debe saber cuál es la siguiente Instruccion a ejecutar. Hay que tener en cuenta que no siempre la siguiente Instruccion es la siguiente posicion de la LinkedList donde almacenamos las intrucciones, sino que el programa se puede encontrar dentro de un bucle, es decir, ejecutando una serie de Instrucciones entre Rep y End, y por lo tanto, al llegar al final, necesita saber cuantas iteraciones ha hecho para saber si debe salir de ese bucle o aun tiene que recorrerlo otra vez. En el caso de que tenga que recorrerlo otra vez, le enviariamos la siguiente instruccion al Rep correspondiente al bucle.



En nuestro código, hemos tenido en cuenta que una lista de instrucciones puede tener varios bucles, y además, que el programa no tiene porque empezar con un bucle.

Hemos visto que Program tiene un atributo llamado *this.loopIteration*. Este guarda la iteración del bucle más interno en el que nos encontramos actualmente. Por lo tanto no tenemos ninguna manera de guardar la informacion de los bucles anteriores. Por ejemplo.



La variable this.LoopIteration, contiene el número de veces que hemos entrado en el bucle actual. Si nos encontramos en la linea 1, cuando loopiteration = 2, es decir cuando ya hemos recorrido el bucle principal la primera vez y aun nos queda recorrerlo por segunda, al entrar a la línea 3, si sobreescribimos loopiteration perdemos la información de cuantas veces hemos recorrido el bucle principal.

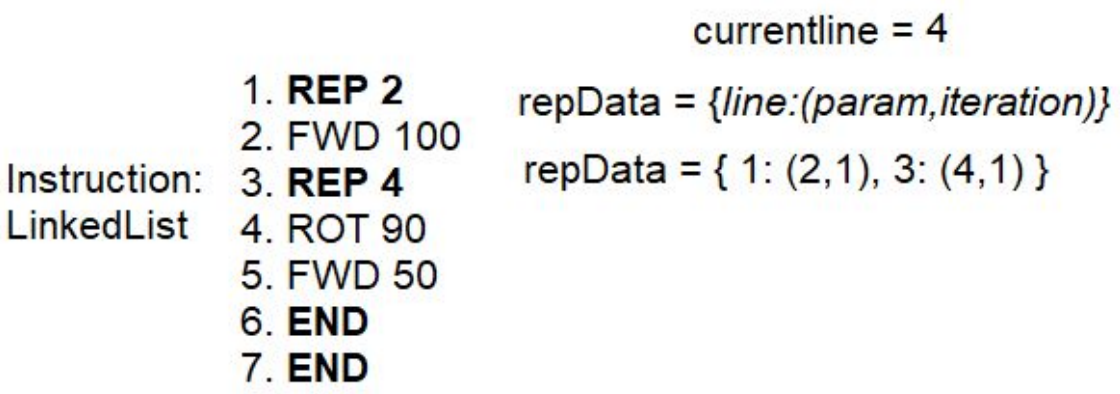
Como solución hemos decidido crear una *HashTable* como atributo de Program, de esta manera podremos guardar la información de cada bucle que tenga el programa; la *HashTable* tendrá como **clave** el nº de línea donde empieza un bucle, y como **valor** una tupla donde el primer elemento es un entero que indica el número de veces que tenemos que iterar ese bucle y el segundo elemento es un entero que indica el número de veces que lo hemos iterado. La llamaremos **repData.**

nº de línea
Nos indica en que linea empieza ese bucle, de esta manera cuando tengamos que devolver la siguiente instrucción, si aún no ha acabado el bucle en el que nos encontramos, sabremos hacia que linea debe ir el programa, devolviendo la siguiente instrucción al último rep.

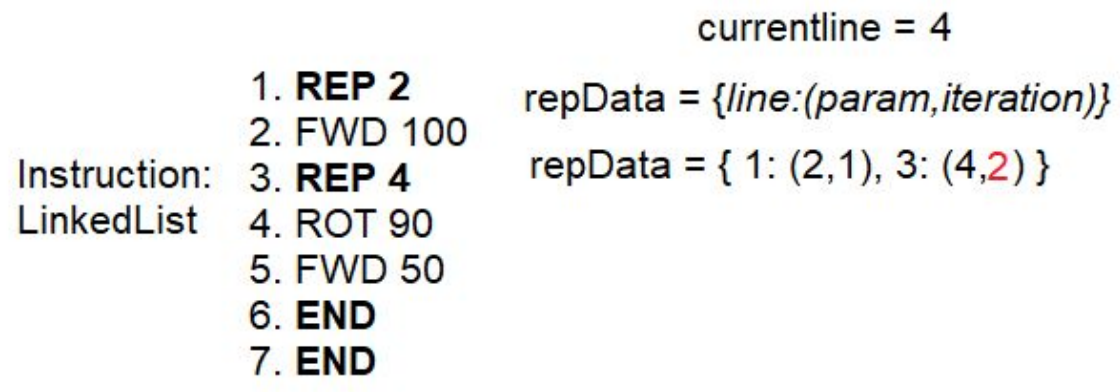
nº repeticiones
Es el parámetro de la instrucción rep, para poder comprobar con el nº de iteración actual si hemos acabado ese bucle.

nº de iteración
Guarda el número de iteración actual sobre un Rep, es decir sobre un bucle, así no perdemos información sobre nuestro número de iteraciones en cada bucle.

Quedaría así, encontrándonos en la línea 4, por primera vez:

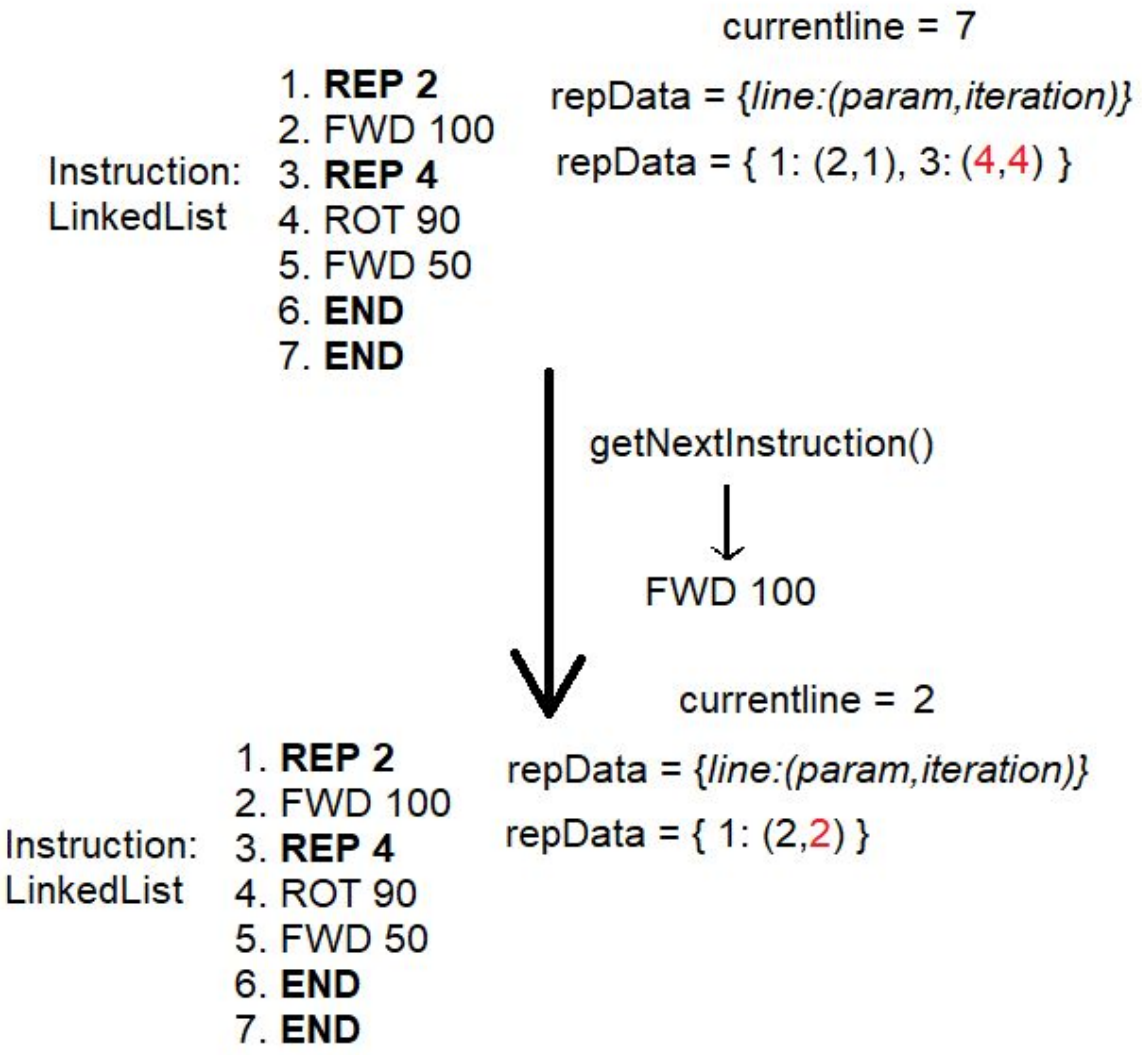


Observamos que el número de iteración de cada bucle está en 1. Una vez llegado a la línea 6, el programa nos vuelve a enviar a la línea 4, quedaría así, estando en la línea 4, por segunda vez.



Una vez hayamos acabado un bucle, lo borramos de RepData, ya que el programa utiliza el **último** elemento de RepData para saber en qué bucle se encuentra, es decir. Comprueba si el primer elemento de la tupla y el segundo son iguales, si no lo son, el programa vuelve a su numero de linea, que lo tenemos guardado en la ultima clave de la HashTable.

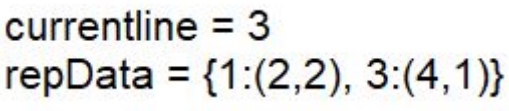
En esta imagen comprobamos que pasa si nos encontramos en la línea 6 y aplicamos getNextInstruction cuando hemos acabado el bucle actual. Es decir el que corresponde a la línea 3.



Observamos que el programa nos manda a la linea 2, y borra el bucle de RepData porque se ha acabado.

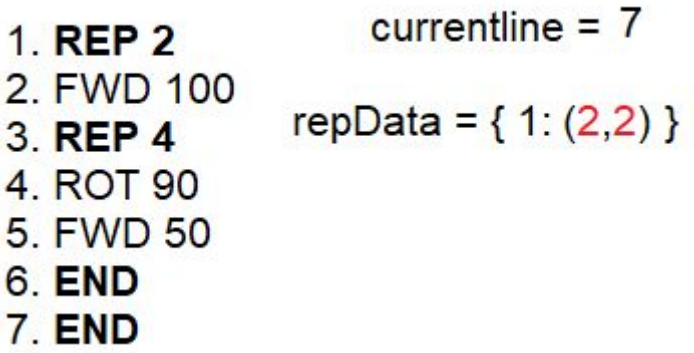
Ahora volveremos a hacer el bucle principal otra vez.

Cuando nos encontremos con el siguiente REP, lo volveremos a añadir a repData, pero no perderemos la información del bucle principal.



Por lo tanto, podemos tener tantos bucles como queramos. Implementando correctamente este método podemos tener cualquier estructura en nuestra lista de Instrucciones.

Aquí nuestro programa llega a su ultima instrucción.



Recordando el diagrama principal (primera pagina), lo primero que comprobamos antes de dar la siguiente instrucción es isFinished(), por lo tanto isFinished() comprobará que nos encontramos en la última línea, es decir, que currentline es igual al tamaño de nuestra lista de instrucciones, y que además repData está vacío, porque ya no quedan bucles por recorrer.