Intérprete de un lenguaje para ANSYS
(Examen septiembre 2016)

En el campo de la ingeniería existen diferentes herramientas de SW para calcular el comportamiento de estructuras (o fluidos) mediante la teoría de los elementos (o volúmenes) finitos. ANSYS es una de esas herramientas y permite usar un lenguaje de programación para automatizar diversas tareas (a través de línea de comandos)

Se desea implementar mediante JFlex y Cup un intérprete del lenguaje usado por ANSYS al que llamaremos **AnsysPL** pero limitando y variando ligeramente sus funciones a un núcleo básico.

NOTA: Para una descripción más detallada puede usarse la guía de ANSYS disponible en http://148.204.81.206/Ansys/150/ANSYS%20Parametric%20DesignLanguage%20Guide.pdf aunque el lenguaje aquí descrito no es exactamente igual (existen pequeñas diferencias).

Aspectos léxicos

Un programa en **AnsysPL** se compone de una o más **sentencias**, cada una de ellas escrita en una línea. El uso de espacios o tabuladores no tiene efecto (el sangrado de las líneas tampoco).

AnsysPL usa parámetros para almacenar información de forma similar a lo que la mayoría de los lenguajes de programación llaman "variables"

El tipo de datos de los parámetros no se declara de forma explícita. Puede almacenar valores numéricos o textuales. Para los números, ya sean reales o enteros, se almacenan como reales (doble precisión). Los textos, como cadenas de caracteres, también se pueden guardar en los parámetros, pero tienen una limitación de ocho caracteres como máximo.

Los nombres admitidos para los parámetros deben empezar con una letra, contener únicamente letras, números o el guión bajo ("_") y no pueden tener una longitud superior a 32 caracteres. De no cumplir estos requisitos, se debería identificar como un error.

Existen nombres no recomendados para los parámetros, que en caso de que no se cumplan deben identificarse con un aviso (warning), aunque no detendrían la ejecución del intérprete. Estas recomendaciones son:

a) Los nombres ARG1 a ARG9 y de AR10 a AR99 están reservados como parámetros locales del entorno;
b) No se debe iniciar el nombre con el guión bajo ("_").

Para incluir comentarios se usa el signo de exclamación (!) que deshabilita todo el texto que le sigue en la línea en la que aparece.

- Sentencias de asignación

La asignación de valores a los parámetros se puede hacer de dos formas alternativas:

1) usando el operador *SET, que indica el nombre de la variable a la que se le asigna el valor (separando por comas los elementos). Por ejemplo, *SET,RESIST,23.67 asigna al parámetro RESIST (que no ha sido declarado previamente) el valor 23.67
2) usando el operador =, que indica el nombre de la variable a la que se le asigna el valor. Por ejemplo, RESIST=23.67 asigna al parámetro RESIST (que no ha sido declarado previamente) el valor 23.67

En caso de que se use un parámetro que no haya sido definido o inicializado previamente, se creará y se le asignará un valor muy próximo a 0, concretamente el valor constante en java Double.MIN_VALUE (4.9E-324). Por ejemplo, si el parámetro X se define como X=Y e Y no ha sido definido previamente, se creará Y y tanto X como Y tomarán el valor 4.9E-324.

El borrado de parámetros se realiza no asignando valor al parámetro, es decir, dejando en blanco la parte correspondiente al valor asignado. Se pueden usar los dos operadores de asignación *SET o =. Por ejemplo, *SET,V4, o V4= borrarían ese parámetro. Pero la asignación del valor 0 o la de un texto vacío (entre comillas o sin comillas) no conllevarían el borrado, sino la asignación de ese valor.

- Sentencias de estado

El listado de parámetros definidos puede hacerse de forma global o individual. Para ello se usa el operador *STATUS. Si no se indica nada después del operador, se recupera la información global. Si se indica el nombre de un parámetro (separándolo con una coma), se recupera de forma individual. En cualquier caso, al recuperar la información siempre se devuelve el número de parámetros definidos (ver ejemplos).

- Expresiones y funciones aritméticas

Los operadores aritméticos utilizan los siguientes signos +, -, *, / y ** (exponenciación) con la siguiente predecencia y asociatividad:

1. Operaciones entre paréntesis (los más internos primero)
2. Exponenciación (en orden, de derecha a izquierda)
3. Multiplicación y división (en orden, de izquierda a derecha)
4. Unario (como +A o -A)
5. Suma y resta (en orden, de izquierda a derecha)

Por ejemplo, la expresión Y2=A+B**C/D*E primero evaluaría B**C, después /D, después *E y terminaría con +A.

El lenguaje también cuenta con algunas funciones paramétricas:

ABS(x)	valor absoluto de x.
SIGN(x,y)	valor absoluto de x con el signo de y. (si y=0, se usa el signo positivo)

- Sentencias de control

Como sentencias de control se pide implementar la de selección. Usa el comando *IF y su sintaxis es:

*IF, *val1*, *oper*, *val2*, *base*

en donde:

- *val1* es el primer valor numérico (o parámetro numérico) en la comparación.
- *oper* es el operador de comparación, uno de estos:
  - EQ: igualdad (*val1* = *val2*).
  - NE: desigualdad (*val1* ≠ *val2*).
  - LT: menor que (*val1* < *val2*).
  - GT: mayor que (*val1* > *val2*).
  - LE: menor o igual que (*val1* ≤ *val2*).
  - GE: mayor o igual que (*val1* ≥ *val2*).
  - ABLT: se aplica ABS a *val1* y *val2* antes de la comparación LT.
  - ABGT: se aplica ABS a *val1* y *val2* antes de la comparación GT.
- *val2* es el segundo valor numérico (o parámetro numérico) en la comparación.
- *base* es la acción a ejecutar si la comparación se evalúa a VERDADERO.
  La opción más común es que *base* tomase el valor THEN (existen otras opciones pero no son objeto de interés en este ejercicio), y por tanto, el comando *IF se convertiría en el inicio de un sentencia if-then-else que se construye de la siguiente forma:

  *IF, *val1, oper, val2*, THEN (necesario)
  ! bloque con instrucciones
  *ELSEIF, *val3, oper, val4* (cero o más ocurrencias)
  ! bloque con instrucciones
  *ELSE (cero o una ocurrencia)
  ! bloque con instrucciones
  *ENDIF (necesario)

Implementación

Se pide implementar un intérprete que lea una lista de sentencias de **AnsysPL** de un fichero de texto con extensión .ans y que dé lugar a la realización de las distintas acciones que se describen mediante ejemplos y se dirigen a la salida estándar, por ejemplo:

```
> java AnsysPL a16.ans
PARAMETER STATUS
( 3 PARAMETERS DEFINED)
NAME VALUE TYPE
v1 -34.56 SCALAR
```

Como resultado de este ejercicio se entregará un fichero comprimido AnsysPL.zip que contenga al menos los ficheros **AnsysPL.jflex**, **AnsysPL.cup** e **AnsysPL.java** más todos aquellos ficheros .java que sean necesarios para la compilación del intérprete mediante la secuencia de instrucciones:

```
> jflex AnsysPL.jflex
> cup AnsysPL.cup
> javac *.java
```

Apartado A

Construir un intérprete que permita utilizar los siguientes operadores:

Entrada	Descripción	Resultado de la ejecución	EJEMPLO
`*SET,RESIST,23.67`	Nombres de parámetros y asignación (*SET o =): almacena en el parámetro el valor indicado (puede ser constante o de otro parámetro)		a01.ans
`*SET,v1,-34.56`			a02.ans
`*SET,TENS,3.07E13`			a03.ans
`*SET,NOMBRE,'ALTO'`			a04.ans
`RESIST=-23.67`			a05.ans
`v1=-34.56`			a06.ans
`APELLIDOS='RODRIGUEZ_LOPEZ'`		ERROR: TEXTO LARGO	a07.ans
`A_PLUS_B=ABC`			a08.ans
`v3=v2`			a09.ans
`3D_X=12`		ERROR: NOMBRE PARÁMETRO	a10.ans
`PI/4=0.7854`		ERROR: NOMBRE PARÁMETRO	a11.ans
`PARAMETRO_SUPERIOR_A_32_CARACTERES=23`		ERROR: NOMBRE PARÁMETRO	a12.ans
`ARG3=0.45`		AVISO: NOMBRE PARÁMETRO	a13.ans
`_TEMP1=12.34`		AVISO: NOMBRE PARÁMETRO	a14.ans

`*SET,RESIST,23.67`
`*SET,v1,-34.56`
`*SET,TEXTO,'hola'`
`*STATUS`
	Impresión (*STATUS): imprime por pantalla	PARAMETER STATUS
		(3 PARAMETERS DEFINED)
		NAME VALUE TYPE
		RESIST 23.67 SCALAR
		v1 -34.56 SCALAR
		TEXTO hola CHARACTER
			a15.ans

`RESIST=23.67`
`v1=-34.56`
`TEXTO='hola'`
`*STATUS,v1`
		PARAMETER STATUS
		(3 PARAMETERS DEFINED)
		NAME VALUE TYPE
		v1 -34.56 SCALAR
			a16.ans

`RESIST=23.67` `*STATUS,Undefined`		PARAMETER STATUS (2 PARAMETERS DEFINED) NAME VALUE TYPE Undefined 4.9E-324 SCALAR	a17.ans
`RESIST=23.67` `*STATUS,PARAMETRO_SUPERIOR_A_32_CARACTERES`		ERROR: NOMBRE PARÁMETRO	a18.ans

`RESIST=23.67`
`*SET,v1,-34.56`
`RESIST=`
`*STATUS,v1`
	Borrado (asignación de "NADA" al parámetro)	PARAMETER STATUS
		(1 PARAMETERS DEFINED)
		NAME VALUE TYPE
		v1 -34.56 SCALAR
			a19.ans

`TXT='texto'`
`*SET,v1,-34.56`
`TXT=`
`*STATUS,TXT`
		PARAMETER STATUS
		(2 PARAMETERS DEFINED)
		NAME VALUE TYPE
		TXT 4.9E-324 SCALAR
			a20.ans

`RESIST=23.67 !resistencia`
`v1=-34.56 !velocidad caso 1`
`TEXTO='hola' !texto para el fichero`
`*STATUS,v1 !imprimir parametros`
	Comentario en línea (!)	PARAMETER STATUS
		(3 PARAMETERS DEFINED)
		NAME VALUE TYPE
		v1 -34.56 SCALAR
			a21.ans

`RESIST=23.67 !resistencia`
`v1=-3!4.56 !velocidad caso 1`
`TEXTO='hola' !texto para el fichero`
`*STATUS,v1 !imprimir parametros`
		PARAMETER STATUS
		(3 PARAMETERS DEFINED)
		NAME VALUE TYPE
		v1 -3 SCALAR
			a22.ans

`A1=-24`
`ABS1=ABS(A1)`
`*STATUS`
	Funciones paramétricas	PARAMETER STATUS
		(2 PARAMETERS DEFINED)
		NAME VALUE TYPE
		A1 -24 SCALAR
		ABS1 24 SCALAR
			a23.ans

`A1=-24`
`A2=3.14`
`X1=SIGN(A2,A1)`
`X2=SIGN(A2,ABS(A1))`
`*STATUS`
		PARAMETER STATUS
		(4 PARAMETERS DEFINED)
		NAME VALUE TYPE
		A1 -24 SCALAR
		A2 3.14 SCALAR
		X1 -3.14 SCALAR
		X2 3.14 SCALAR
			a24.ans

Apartado B

Construya un intérprete que permita utilizar los siguientes operadores:

Entrada	Descripción	Resultado de la ejecución	EJEMPLO
A1=24 **A2=3+4** **A3=10+A2** ***STATUS**	**Fórmula aritmética: operadores suma (+), resta (-), multiplicación (*), división entera (/), exponenciación (**)**	PARAMETER STATUS (3 PARAMETERS DEFINED) NAME VALUE TYPE A1 24 SCALAR A2 7 SCALAR A3 17 SCALAR	b01.ans
A1=23** **A2=A1*2** **A3=(A1+A2)/2** **A4=A3/A1** ***STATUS**		PARAMETER STATUS (4 PARAMETERS DEFINED) NAME VALUE TYPE A1 8 SCALAR A2 16 SCALAR A3 12 SCALAR A4 1.5 SCALAR	b02.ans
A=2 **B=4** **C=5** **D=10** **E=7** **X=A+B**C/D*E** ***STATUS,X**		PARAMETER STATUS (6 PARAMETERS DEFINED) NAME VALUE TYPE X 718.8 SCALAR	b03.ans
A=2 **B=4** **C=5** **D=10** **E=7** **X=A+B**C/D*E** **Y=X*Z** ***STATUS,Y**		PARAMETER STATUS (8 PARAMETERS DEFINED) NAME VALUE TYPE Y 5.67033E-28 SCALAR	b04.ans
A=3 ***IF,A,GT,0,THEN** **SALIDA=-1** ***ENDIF** ***STATUS,SALIDA**	**Sentencia de selección (*IF)**	PARAMETER STATUS (2 PARAMETERS DEFINED) NAME VALUE TYPE SALIDA -1 SCALAR	b05.ans

A=-33
***IF,A,GT,0,THEN**
SALIDA=-1
***ELSE**
SALIDA=-100
***ENDIF**
***STATUS,SALIDA**
		PARAMETER STATUS
		(2 PARAMETERS DEFINED)
		NAME VALUE TYPE
		SALIDA -100 SCALAR
			b06.ans

A=-33
B=100
***IF,A,GT,B,THEN**
SALIDA=1
***ELSEIF,A,LT,B**
SALIDA=-1
***ELSE**
SALIDA=0
***ENDIF**
***STATUS,SALIDA**
		PARAMETER STATUS
		(3 PARAMETERS DEFINED)
		NAME VALUE TYPE
		SALIDA -1 SCALAR
			b07.ans

***IF,3,GT,0,THEN**
  ***IF,-3,LT,0,THEN**
  SALIDA=-1
  ***ENDIF**
***ENDIF**
***STATUS,SALIDA**
		PARAMETER STATUS
		(1 PARAMETERS DEFINED)
		NAME VALUE TYPE
		SALIDA -1 SCALAR
			b08.ans

A=3 ***IF,A,GT,0,THEN** **SALIDA=-1** ***STATUS,SALIDA**		ERROR: SENTENCIA *IF MAL CONSTRUIDA	b09.ans

A=-33
***IF,A,GT,0,THEN**
SALIDA=-1
***ENDIF**
SALIDA=-100
***ELSE**
***STATUS,SALIDA**
		ERROR: SENTENCIA *IF MAL CONSTRUIDA	b10.ans