4º Ingeniería Informática
II26 Procesadores de lenguaje
Control de la práctica de miniint (25 de abril de 2009)

INSTRUCCIONES:

- La duración del examen es de dos horas.
- Antes de empezar, asegúrate de que el usuario con el que estás trabajando coincide con el del recuadro del final de este enunciado.
- Rellena el recuadro con tus datos.
- Ejecuta el programa `./prepara.py` pasándole como parámetro tu DNI. Este creará un fichero con tu DNI y nombre y un directorio llamado `miniint`, donde debes guardar tu intérprete.
- Al introducir el USB, debería montarse automáticamente. Si fuera necesario, puedes montarlo y desmontarlo mediante `mount` y `umount` sobre el directorio apropiado.
- Cuando termines el examen, entréganos esta hoja.

PREGUNTA 1 (1 PUNTO)

El objetivo de este ejercicio es que tu intérprete MINIINT acepte, además de lo que se pedía en la práctica 3, sentencias condicionales múltiples utilizando la nueva palabra reservada `si_no_si`, que tiene el mismo significado que `elif` en Python. Para ello, sustituye en el esquema de traducción la producción correspondiente al `si` por la siguiente[1]:

```
1  <Sentencia> ->
2    si <Expresion> entonces <Sentencia>
3      @expresiones= [Expresion1.arb]@
4      @sentencias= [Sentencia1.arb]@
5    (
6      si_no_si <Expresion> entonces <Sentencia>
7        @expresiones.append(Expresion2.arb)@
8        @sentencias.append(Sentencia2.arb)@
9    )*
10   @sentenciaSino=None@
11   ( si_no <Sentencia>
12       @sentenciaSino= Sentencia3.arb@
13   )? fin
14   @Sentencia.arb= AST.NodoSiMultiple(expresiones, sentencias, sentenciaSino, si.nlinea)@
15   ;
```

y añade `"si_no_si"`, a la lista de palabras reservadas que se encuentra en el mismo fichero. Tras modificar ese fichero, debes compilarlo de nuevo utilizando, por ejemplo, `../metacomp/metacomp miniint.mc -s miniint`. Sin modificar nada más en el esquema de traducción, debes implementar lo necesario para poder ejecutar programas en los que se utilicen ese tipo de sentencias, teniendo en cuenta lo siguiente:

1. Se debe ejecutar únicamente la sentencia asociada a la primera condición que resulte ser cierta o la asociada al `si_no` final en caso de que exista y ninguna condición sea cierta.
2. Se debe comprobar que las expresiones tanto del `si` como de los `si_no_si` son de tipo lógico. Si no es así, el mensaje que debes pasar a `errores.semantico` es exactamente `"Error de tipos en condicional múltiple."`.
3. Se deben poder utilizar condicionales múltiples dentro de otras condicionales, de forma anidada.
4. Se deben poder utilizar condicionales múltiples dentro de funciones. En ese caso, la ejecución de una sentencia `devuelve` mientras se ejecuta la sentencia condicional debe tener el comportamiento habitual, finalizando la ejecución de la función.

[1] Tienes esta producción en el fichero `produccionSi.mc` de tu *home*.

El siguiente ejemplo (del que tienes una copia en el fichero `ejemplo.min` de tu *home*) ilustra algunos de estos aspectos:

```
1  globales
2    basura: entero;
3  fin
4
5  funcion califica(n: entero): cadena
6  es
7  secuencia
8    si n< 0 entonces
9      devuelve "negativo";
10   si_no_si n= 0 entonces
11     devuelve "cero";
12   si_no
13     devuelve "positivo";
14   fin
15 fin
16
17 funcion describe(n: entero): entero
18 es
19   secuencia
20     escribe "El número "; escribe n; escribe " es ";
21     escribe llama califica(n); escribe ".";  nl;
22     devuelve 0;
23   fin
24
25 secuencia
26   basura:= llama describe(0-1);
27   basura:= llama describe(1-1);
28   basura:= llama describe(2-1);
29 fin
```

Al ejecutar este ejemplo con tu solución, la salida debe ser la siguiente:

```
El número -1 es negativo.
El número 0 es cero.
El número 1 es positivo.
```

Diseña tú mismo las pruebas adicionales que consideres conveniente.

Importante:

- Guarda los ficheros del intérprete, una vez modificados, en el directorio `miniint` de tu *home*.
- El fichero principal debe llamarse `miniint` y ser ejecutable.
- Utiliza el *script* `verifica.sh` de tu directorio *home* para comprobar que la estructura es la que pedimos. **Se penalizarán los exámenes para los que este script dé errores.**