UNIVERSITAT JAUME·I

4º Ingeniería Informática

II26 Procesadores de lenguaje

Control de la práctica de minicomp (30 de mayo 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 `minicomp`, donde debes guardar tu compilador.
- 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 (1,5 PUNTOS)

El objetivo de este ejercicio es que tu compilador MINICOMP acepte, además de las modificaciones que se pedían en la práctica 4, asignaciones de un valor a un rango de posiciones en un vector con los requisitos que se especifican a continuación:

1. Utilizaremos la sintaxis siguiente:

   ```
   vector ( inferior : superior ) <-- valor ;
   ```

   donde:

   - *vector* identifica al vector y puede ser tanto un identificador (por ejemplo, `v`) como un acceso a una componente de un vector (por ejemplo, `v[4]`); en cualquier caso, su tipo base debe ser elemental;
   - *inferior* y *superior* son dos expresiones de tipo entero;
   - y *valor* es una expresión que puede o bien ser del mismo tipo que el tipo base de *vector* o bien ser de tipo entero si el tipo base de *vector* es el tipo real.

2. La construcción completa constituye una sentencia.

3. Los posibles efectos secundarios de evaluar las expresiones ocurren sólo una vez, aunque no está especificado el orden de evaluación.

4. Llamemos v al vector referenciado por *vector*, i al resultado de evaluar *inferior*, j al de *superior* y e al de *valor*. El efecto de la sentencia es que inmediatamente después de su ejecución $v[i], v[i+1], \ldots, v[j]$ tienen el valor e. En caso de que el tipo base de v fuera real y e entero, se produciría una promoción implícita de entero a real. Si $i > j$, la sentencia no tiene más efectos que los posibles efectos secundarios de evaluar *inferior*, *superior* y, posiblemente, *valor*. Está indefinido el comportamiento del programa en caso de que i sea menor que cero o j mayor o igual que el número de elementos de v.

Por ejemplo, suponiendo las declaraciones

```
v: vector [30] de real;
m: vector [10] de vector[20] de vector[30] de real;
```

dos sentencias válidas serían:

```
v(11:22) <-- 33;
m[4+4][5-5](6+6:7+7) <-- llama AreaTriangulo (88, 99);
```

la primera rellena las componentes 11 a 22 de `v` con un 33 (convertido a real), la segunda rellena los componentes 12 a 14 de `m[8][0]` con el resultado de la llamada a `AreaTriangulo`, que se realiza sólo una vez.

Puedes comprobar fácilmente si *`tipo`* es un vector con tipo base elemental usando la siguiente expresión:

`tipo``.nombre== "Array" and `*`tipo`*`.base.elemental()`

Para que tengas una primera prueba inicial de tu compilador, hemos dejado en el *home* de tu usuario los ficheros `ejemplo1.i` y `ejemplo2.i` junto con los correspondientes `.o` y `.e`. Obviamente, debes preparar tus propias pruebas para verificar que tu compilador se comporta correctamente también en aquellos aspectos no abarcados por las nuestras.

Recuerda: los ficheros de tu compilador deben estar en el directorio `minicomp` del *home* de tu usuario. El fichero principal debe llamarse `minicomp` y ser ejecutable. Puedes utilizar 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.**