4º Ingeniería Informática
II26 Procesadores de lenguaje
Examen (24 de junio de 2008)

PREGUNTA 1 (5 PUNTOS)

Explica cómo debería modificarse la versión 7 de `minicomp` (la que incluye las modificaciones hasta "promociones implícitas de entero a real", inclusive) de modo que aceptara las extensiones que se presentan a continuación. Las extensiones son independientes entre sí; no hace falta que consideres sus posibles interacciones.

Procura que tu descripción sea clara, escueta y precisa. Para ello, puede facilitarte la exposición una estructura que siga las distintas etapas del compilador.

Puedes optar por descripciones algorítmicas o en lenguaje natural para lograr una mayor sencillez en la explicación, pero dejando claro qué se modificaría y cómo. Como mínimo: las descripciones de modificaciones del nivel léxico deben explicitar, si las hay, las nuevas expresiones regulares y los atributos asociados a posibles nuevas categorías; las de modificaciones del nivel sintáctico o semántico deben aclarar las reglas, acciones semánticas y/o métodos de comprobación afectados; las de modificaciones de la generación de código deben dejar claro qué instrucciones de máquina virtual se emitirían con la modificación. Por ejemplo: sería aceptable una frase como "se comprobaría que el hijo izquierdo es de tipo entero" (es trivial transformar la comprobación en un test), pero no una como "se generaría código para sumar los dos números" (no sabemos si se generaría una instrucción o varias, cuál o cuáles serían, dónde están los números —¿en registros, en memoria, en la pila?—, etc.).

Explicita cualquier asunción que hagas acerca del enunciado propuesto.

E1. Producto escalar de vectores (3 puntos)

Esta extensión permite calcular el producto escalar de dos vectores[1], u y v, empleando la notación `<u,v>`. Tanto u como v deben ser vectores de reales de la misma talla, sin que haya promociones de vector de enteros a vector de reales. El resultado es de tipo real. Este operador puede aparecer como operando en expresiones mayores como por ejemplo:

```
escribe 2 + <u,v>*3 + <u,v>*<v,w>;
```

Ten en cuenta que cualquiera de los vectores que se están multiplicando puede ser un componente de un vector mayor, como en `<m[4],v>`, donde `m` es un `vector[5] de vector[3] de real` y `v`, un `vector[3] de real`. Si en el acceso a la componente de un vector se producen efectos secundarios, estos suceden sólo una vez. Así, en `<m[llama f()],v>`, la función `f` es llamada una sola vez.

El número de instrucciones generadas debe ser independiente del tamaño de los vectores.

E2. Sentencia de escritura múltiple (2 puntos)

Mediante esta extensión se permite unir varias sentencias de escritura en una sola. Para ello, se emplea el pseudooperador `<+>` de modo que una sentencia como:

```
escribe "2+2=" <+> 2+2 <+> nl;
```

escribiría en primer lugar la cadena `2+2=`, después `4` y, por último, un fin de línea.

Más concretamente, una sentencia de escritura se compondrá de:

- La palabra reservada `escribe`.
- Una secuencia de una o más expresiones de tipo entero, real o cadena, separadas por el símbolo `<+>`. Tras esta secuencia pueden aparecer opcionalmente un símbolo `<+>` y la palabra reservada `nl`.
- Un punto y coma.

Observa que `<+>` es un único componente léxico.

Si quieres alcanzar la máxima nota, no debes crear nuevos tipos de nodos ni modificar los existentes.

[1] El producto escalar de dos vectores u y v de talla n es igual a $\sum_{i=0}^{n-1} u[i]v[i]$.

Pregunta 2 (2 puntos)

Escribe tanto una expresión regular como un AFD para el lenguaje de los comentarios con las siguientes características:

1. Empiezan por `<<` y terminan por `>>`.
2. En su interior admiten fragmentos especiales que empiezan y terminan por comillas simples `'`.
3. En el interior de un comentario las comillas `'` sólo pueden aparecer en una cantidad par, delimitando los fragmentos especiales.
4. En el interior de un comentario no se admite la secuencia `>>`, excepto dentro de los fragmentos especiales.

Esto son ejemplos de comentarios válidos:

```
<<>>
<< abc >>
<< abc 'de' fg 'hijk' lm >>
<< '' abc 'de' fg 'hi>>jk' '>>>>>' l>m >>
```

Y esto no son comentarios válidos:

```
<< abc 'def >>
<< abc >> def >>
```

Pregunta 3 (1,5 puntos)

Escribe las producciones de una gramática $G = (\{\langle A\rangle, \langle B\rangle, \langle C\rangle\}, \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}, P, \langle A\rangle)$ sabiendo que es LL(1), que todos los no terminales son anulables y que las únicas celdas no vacías de su tabla de análisis son las sombreadas en el esquema siguiente:

	a	b	c	$
⟨A⟩	■	■	■	■
⟨B⟩		■	■	
⟨C⟩			■	

Pregunta 4 (1,5 puntos)

Supongamos que `escribeMayores` es una sentencia que escribe los elementos de una lista que son mayores que su primer parámetro. Por ejemplo, la sentencia

```
escribeMayores (2+3, (4; 5+6+7; 8; 2));
```

escribiría los valores `18` y `8`.

La siguiente gramática modela la sentencia `escribeMayores`:

⟨Sentencia⟩ → **escribeMayores abreParéntesis** ⟨Expresión⟩ **coma** ⟨Lista⟩ **cierraParéntesis puntoYComa**
⟨Lista⟩ → **abreParéntesis** ⟨Expresión⟩⟨MásExpresiones⟩ **cierraParéntesis**
⟨MásExpresiones⟩ → **puntoYComa** ⟨Expresión⟩⟨MásExpresiones⟩ | λ
⟨Expresión⟩ → **número** ⟨MásNúmeros⟩
⟨MásNúmeros⟩ → **suma número** ⟨MásNúmeros⟩ | λ

Añade a esta gramática (sin modificarla de ninguna forma) las acciones semánticas que ejecuten `escribeMayores`. Asume que los componentes de la categoría `número` son enteros y que su valor está en el atributo v. Sólo puedes utilizar atributos de tipo entero o lógico (no está permitido usar, por ejemplo, listas o variables globales). Además, no puedes utilizar como heredados atributos sintetizados y viceversa.

Indica para cada atributo si es heredado o sintetizado y qué representa.

Duración del examen: 4 horas