PROCESADORES DE LENGUAJES
4º Ingeniería Informática
Convocatoria Extraordinaria de Septiembre
1-Septiembre-2006

DNI:		**Apellidos y Nombre:**		**Grupo:**	
Profesor de Prácticas:					

Observaciones: Este examen es obligatorio para aquellos que ya han presentado y aprobado las prácticas de laboratorio en este curso 2005/2006 (se consideran superadas las prácticas de laboratorio con una calificación igual o superior a 1.5 puntos). La calificación total del apartado de prácticas se considera un factor entre 0 y 1 a multiplicar por la calificación total obtenida en las prácticas de laboratorio.

PRÁCTICAS

1. Dado el lenguaje asignado para la realización de las prácticas, se va a añadir como tipo básico el **número complejo** con las operaciones de **suma**, **resta**, **multiplicación** y **división** entre complejos (con la misma precedencia que los operadores empleados para expresiones de tipo entero y real), los operadores unarios **mas** y **menos** (con la misma precedencia que los unarios ya existentes) y operadores unarios para la extracción de la **parte real** y la **parte imaginaria** (ambos con la máxima prioridad que el resto de los operadores que existan).

   Determinar todos los cambios necesarios en el traductor a nivel de:

   (a) **Análisis Léxico:** Determinar qué lexemas y/o tokens nuevos surgen, en su caso, y especificarlos en sintaxis de LEX.

   (b) **Análisis Sintáctico:** Añadir reglas gramaticales para permitir esta nueva funcionalidad, en su caso, y especificarlas usando la sintaxis de YACC.

   (c) **Análisis Semántico:** Añadir las acciones semánticas correspondientes a estas reglas y especificarlas usando la sintaxis de YACC.

Solución:

Análisis Léxico

En primer lugar aparece un nuevo tipo de dato dentro de los cuatro básicos que denominaremos `complejo` para aquellos que tengan palabras reservadas en castellano y `complex` para los que no. No se crea un nuevo token sino que se añade un nuevo lexema al token `TIPO`. El valor del atributo lo define la constante `ATR_TIPO_COMPL`, que se inicializará al siguiente valor de los atributos posibles de un tipo.

Por otro lado, los operadores que permiten definir el álgebra de tratamiento de este tipo de dato serán sobrecargados a excepción de los unarios que extraen la parte real y la parte imaginaria. Los operadores sobrecargados serán los de multiplicación, división, suma y resta entre dos datos de tipo complejo y la suma y resta unaria. Aparecerán dos nuevos operadores que definimos así:

- ($) Operador unario que devuelve la parte real de un número complejo.
- (#) Operador unario que devuelve la parte imaginaria de un número complejo.

Al tener estos operadores unario otorgada la mayor de las precedencias, no sería posible agruparlos dentro del token de operadores unarios ya que éstos tendrían menos precedencia que los nuevos. Por lo tanto, al tener la misma misión sintáctica, asociatividad y precedencia, creamos el nuevo token denominado `OP_COMPL_UNA`

La descripción en sintaxis de LEX quedaría así:

```
.....
%%
.....
"complex"     { yylval.atrib= ATR_TIPO_COMPL ; return TIPO ; }
.....
"$"           { yylval.atrib= ATR_REAL_COMPL ; return OP_COMPL_UNA ; }
"#"           { yylval.atrib= ATR_IMAG_COMPL ; return OP_COMPL_UNA ; }
%%
.....
```

donde `ATR_REAL_COMPL` y `ATR_IMAG_COMPL` son dos constantes enteras de distinto valor que sirven para dar valor al atributo del token `OP_COMPL_UNA` e `yylval` es la variable de LEX/YACC de tipo `YYSTYPE`, definida con la misma estructura que la práctica de análisis semántico. En dicha variable existe un campo denominado "`atrib`"que es de tipo entero que almacena valores referentes al atributo del token (de cara a distinguir los lexemas en el análisis semántico).

Obviamente, es posible definir estos nuevos operadores con otros caracteres, en el caso de que éstos hubiesen sido definidos para otro propósito.

Análisis Sintáctico

Se debe definir la asociatividad y precedencia del nuevo token `OP_COMPL_UNA` que debe ser a la derecha y situado al final de todos para otorgarle la mayor precedencia.

En cuanto a producciones, la única regla gramatical que se debe añadir sería la que permite la extracción de la parte real y la parte imaginaria de una expresión formada por un tipo complejo. En sintaxis de YACC sería así:

```
.....
%left  ....
.....
%right OP_UNA
%right OP_COMPL_UNA

%%
.....
Expresion : ...
          | OP_COMPL_UNA Expresion
          | ...
;
.....
%%
.....
```

Análisis Semántico

Dado que ya existen las reglas sintácticas que operan con expresiones unarias y binarias para los operadores aritméticos binarios y unarios, a nivel semántico habría que considerar el tratamiento cuando las expresiones sean de tipo complejo. En cuanto a la regla sintáctica que aparece nueva, también tendríamos que realizar la pertinente consulta para determinar si el operador correspondiente tiene asociada una expresión de tipo complejo.

En sintaxis de YACC sería así:

```
.....
%%
.....
expresion : .....
          | expresion OP_MUL_DIV_BIN expresion
              {
                 .....
                 // Tipo Complejo en expresión
                 if ($1.tipo==ATR_TIPO_COMPL && $1.tipo==$3.tipo)
                    $$.tipo= $1.tipo ;
                 else
                 {
                    $$.tipo= incorrecto ;
                    printf ("Error semántico:
                             tipos incompatibles en dato complejo\n");
                 }
                 .....
              }
          | expresion OP_SUM_RES_UBI expresion
              {
                 .....
                 // Tipo complejo en expresión
                 if ($1.tipo==ATR_TIPO_COMPL && $1.tipo==$3.tipo)
                    $$.tipo= $1.tipo ;
                 else
                 {
                    $$.tipo= incorrecto ;
                    printf ("Error semántico:
                             tipos incompatibles en dato complejo\n");
```

```
                }
                .....
            }
        | OP_SUM_RES_UBI expresion %prec OP_UNA
            {
                .....
                // Tipo complejo en expresión
                if ($2.tipo==ATR_TIPO_COMPL)
                   $$.tipo= complejo ;    // Se sintetiza un dato complejo.
                .....
            }
        | OP_COMPL_UNA expresion
            {
                if ($2.tipo=ATR_TIPO_COMPL)
                   $$.tipo= real ;       // Se sintetiza un dato de tipo float.
                else
                {
                   $$.tipo= incorrecto ;
                   printf ("Error semántico:
                            tipo incorrecto en dato complejo\n");
                }
            }
        | .....
%%
.....
```