LENGUAJES DE PROGRAMACIÓN
Y
PROCESADORES DE LENGUAJES

Construcción de un compilador

`MenosC`

Parte-II: comprobaciones semánticas

Material auxiliar de prácticas

- `Makefile`. Una nueva versión.
- `principal.c`. Una nueva versión en el directorio **src**.
- `libtds`. Librería con las operaciones para la manipulación de la Tabla de Símbolos

  `libtds.h`, el fichero de cabecera, en el directorio **include**;

  `libtds.a`, la librería, en el directorio **lib**.
- *Programas de prueba*.

Especificación Semántica

- En el compilador solo se usan constantes enteras. Si el analizador léxico encuentra una constante real se debe devolver su valor entero truncado.
- Todas las variables deben declararse antes de ser utilizadas.
- La talla de los tipos *entero* y *lógico* se debe definir en TALLA_TIPO_SIMPLE= 1.
- El tipo lógico `bol` se representa numéricamente como un entero: con el valor 0, para el caso `falso`, y 1, para el caso `verdad`.
- No existe conversión de tipos entre `int` y `bool`.
- Los argumentos del operador módulo, " %", deben ser enteros.
- Los índices de los vectores van de `0` a `cte-1`, siendo `cte` el número de elementos, que debe ser un entero positivo.
- No es necesario comprobar los índices de los vectores en tiempo de ejecución.
- Las expresiones de las instrucciones `if-elseif-else`, `while` y `do-while` deben ser de tipo lógico.
- Por defecto las restricciones semánticas serán las propias del lenguaje ANSI C.

➢ Estructura de la TDS

Constantes, variables globales y estructuras básicas (ver Sección 5.2 del Enunciado)

➢ Funciones de manipulación de la TDS

```
int insertarTDS (char *nom, int tipo, int desp, int ref) ;
/* Inserta en la TDS toda la información asociada con un objeto definido por
   el usuario: nombre ‘‘nom’’; tipo ‘‘tipo’’; desplazamiento relativo en el
   segmento de variables ‘‘desp’’ y referencia ‘‘ref’’ a una posible subtabla
   de vectores. Donde, ‘‘ref = -1’’, para los objetos de tipo simple. Si el
   objeto ya existe devuelve el valor ‘‘FALSE = 0’’ (‘‘TRUE = 1’’ en caso
   contrario).  */

SIMB obtenerTDS (char *nom) ;
/* Obtiene toda la información asociada con un objeto de nombre ‘‘nom’’ y
   la devuelve en una estructura de tipo ‘‘SIMB’’ (ver libtds.h). Si
   el objeto no está declarado, en el campo ‘‘tipo’’ devuelve ‘‘T_ERROR’’.  */
```

Tabla de Símbolos

```
int insertaTDArray (int telem, int nelem) ;
/* Inserta en la Tabla de Arrays la información de un array cuyos elementos
   son de tipo ‘‘telem’’ y el número de elementos es ‘‘nelem’’. Devuelve su
   referencia en la Tabla de Arrays.  */

DIM obtenerInfoArray (int ref) ;
/* Obtiene la información de un array referenciado por ‘‘ref’’ en la Tabla
   de Arrays y la devuelve en una estructura de tipo ‘‘DIM’’ (ver libtds.h).  */

void mostrarTDS () ;
/* Muestra toda la información de la TDS.  */
```

➢ Ejemplo de comprobaciones de tipo en *declaraciones*

```
declaracion | tipoSimple ID_ ACOR_ CTE_ CCOR_ PUNTOCOMA_

            { int numelem = $4; int refe;
              if ($4 <= 0)  {
                yyerror("Talla inapropiada del array");
                numelem = 0;
              }
              refe = insertaTDArray($1, numelem);
              if ( ! insertarTDS($2, T_ARRAY, dvar, refe) )
                yyerror ("Identificador repetido");
              else dvar += numelem * TALLA_TIPO_SIMPLE;
            }
```

➢ Ejemplo de comprobaciones de tipo en la *asignación*

```
expresion
   | ID_  ASIGNACION_  expresion  PUNTOCOMA_

     { SIMB sim = obtenerTDS($1); $$.tipo = T_ERROR;

       if (sim.tipo == T_ERROR) yyerror("Objeto no declarado");
       else if (! ((sim.tipo == $3.tipo == T_ENTERO) ||
                   (sim.tipo == $3.tipo == T_LOGICO)))
         yyerror("Error de tipos en la 'instrucción de asignación'");
       else $$.tipo = sim.tipo;
     }
```

† Advertid que este código se debería modificar para que solo de un nuevo mensaje de error si el error se produce en esta regla, y no si proviene de errores anteriores a través de $1 o $3.