Declaración y uso de identificadores

Descripción general

- **Descripción de la semántica:** los identificadores tienen que ser declarados antes de su uso y deben ser únicos en su ámbito. Es decir:
  - Las **variables (globales y locales)** deben ser declaradas antes de su uso y además ser únicas en su ámbito de declaración.
  - Las **funciones** deben ser declaradas antes de su uso y ser únicas en su ámbito de declaración.
  - Los **parámetros** formales de las funciones son tratados como variables locales de la función y como tales deben ser únicos en su ámbito de declaración. Son declarados implícitamente antes de su uso ya que aparecen en la cabecera de la función.

- **Comprobaciones semánticas:** se necesitan dos acciones diferentes:
  - En la **declaración** de una variable/función/parámetro hay que comprobar que no existe dentro de su ámbito otra variable/función/parámetro con el mismo identificador antes de registrarla en el sistema, es decir, guardarla en la tabla de símbolos.
  - En el **uso**/aparición de una variable/función/parámetro hay que comprobar que previamente ha sido declarada, es decir, que está guardada en la tabla de símbolos.

Declaración y uso de identificadores

Gestión de ámbitos

- La gestión de ámbitos anidados se puede implementar utilizando una pila de tablas de símbolos. Si el número máximo de ámbitos es 2, también se puede usar 2 tablas para gestionar los ámbitos.

- En la compilación, es necesario realizar las siguientes acciones:

  - **Antes de la primera declaración del programa fuente se debe inicializar la tabla de símbolos**. Este proceso de inicialización implica crear una tabla de símbolos vacía correspondiente al ámbito principal. La inicialización se puede realizar en uno de los dos puntos siguientes:
    - En el programa principal antes de invocar a la función de análisis sintáctico.
    - En la producción del axioma antes de la sección de declaraciones. En cualquiera de los puntos •

      programa: • TOK_MAIN • '{' • declaraciones funciones sentencias '}'

      Se crea un nuevo no terminal "ficticio" y se añade una producción lambda en cuya acción semántica se inicializa la tabla de símbolos. Por ejemplo:

```
programa: inicioTabla TOK_MAIN '{' declaraciones funciones sentencias '}'
inicioTabla: {
                /* Acciones de inicialización de la tabla de símbolos */
             }
```

Declaración y uso de identificadores

Gestión de ámbitos

- **En la declaración de una función hay que abrir un nuevo ámbito**, el correspondiente a la función. Se verá con detalle más adelante.

- **Al terminar la compilación**, liberar los recursos de la tabla de símbolos. Se puede hacer en:
  - El programa principal, cuando termina la función de análisis sintáctico.
  - En el fichero de especificación Bison. En este caso habría que contemplar las dos posibles finalizaciones de la función de análisis, con error o sin error. Si no hay error se puede hacer la liberación en la producción del axioma, al final de la misma. Si hay error se podría hacer en la función yyerror.

Declaración y uso de identificadores

Inserción en la tabla de símbolos

- Las declaraciones de **variables globales** y **locales** se ajustan a la sintaxis definida por las siguientes producciones de la gramática de ALFA:

  - (1) programa : TOK_MAIN '{' declaraciones funciones sentencias '}'
  - (2) declaraciones : declaracion
  - (3) | declaracion declaraciones

  - (4) declaracion : clase identificadores ';'

  - (18) identificadores : identificador
  - (19) identificadores : identificador ',' identificadores

  - (22) funcion : TOK_FUNCTION tipo identificador '(' parametros_funcion ')' '{' declaraciones_funcion sentencias '}'

  - (28) declaraciones_funcion : declaraciones
  - (29) |

  - identificador : TOK_IDENTIFICADOR ➡ Esta producción se añadió a la gramática cuando se implementó en analizador sintáctico.

Declaración y uso de identificadores

Inserción en la tabla de símbolos

- Los **parámetros** de las **funciones** se ajustan a la sintaxis definida por las siguientes producciones de la gramática de ALFA:
  - (22) funcion : TOK_FUNCTION tipo identificador '(' parametros_funcion ')' '{' declaraciones_funcion sentencias '}'
  - (23) parametros_funcion : parametro_funcion resto_parametros_funcion
  - (24) |
  - (25) resto_parametros_funcion : ';' parametro_funcion resto_parametros_funcion
  - (26) |
  - (27) parametro_funcion : tipo identificador
  - identificador : TOK_IDENTIFICADOR

- La declaración de **funciones** tiene la siguiente sintaxis:
  - (22) funcion : TOK_FUNCTION tipo identificador '(' parametros_funcion ')' '{' declaraciones_funcion sentencias '}'
  - identificador : TOK_IDENTIFICADOR

Declaración y uso de identificadores

Inserción en la tabla de símbolos

- Del estudio de las producciones relacionadas con la declaración de variables globales y locales, parámetros y funciones, se puede deducir que **cada vez que se realiza una declaración se reduce la producción siguiente**:
  - identificador : TOK_IDENTIFICADOR

  Por lo tanto, **parece que el punto adecuado para insertar los identificadores de las variables globales y locales, parámetros y funciones en la tabla de símbolos es la acción semántica de la producción anterior.** No obstante, veremos que es más adecuado utilizar esta producción únicamente para variables locales y globales, y usar otros mecanismos para la inserción de los identificadores de parámetros y funciones.

Declaración y uso de identificadores

Inserción en la tabla de símbolos

- Independientemente del punto seleccionado para la inserción de un identificador en la tabla de símbolos, es necesario **disponer** en dicho punto de **toda la información asociada al identificador**, como por ejemplo:
  - **nombre**
  - **categoría** (variable, parámetro o función)
  - **clase** (escalar o vector)
  - **tipo** (entero o booleano)
  - **tamaño** (número de elementos de un vector)
  - **número de variables locales**
  - **posición de variable local**
  - **número de parámetros**
  - **posición de parámetro**

- **En el momento de la inserción en la tabla de símbolos se debe comprobar la unicidad** de los identificadores.

Declaración y uso de identificadores

Inserción en la tabla de símbolos

- Los objetivos del análisis semántico en el procesamiento de la **declaración de identificadores** son los siguientes:
  - **Insertar en la tabla de símbolos cada identificador de variable global o local, parámetro o función con su correspondiente información asociada.**
  - **Controlar la unicidad de identificadores** (haciendo uso de la tabla de símbolos).
- Para la inserción de cualquier identificador es necesario definir:
  - El punto en el que se realiza la inserción.
  - Los mecanismos que permiten disponer en ese punto de la información asociada al identificador.
- Se distinguirán tres casos diferentes:
  - Inserción de identificadores de variables globales y locales.
  - Inserción de identificadores de parámetros de funciones.
  - Inserción de identificadores de funciones.

Declaración de variables globales y locales

Introducción

- El punto adecuado para insertar los identificadores de variables globales y locales en la tabla de símbolos es la acción semántica de la producción:

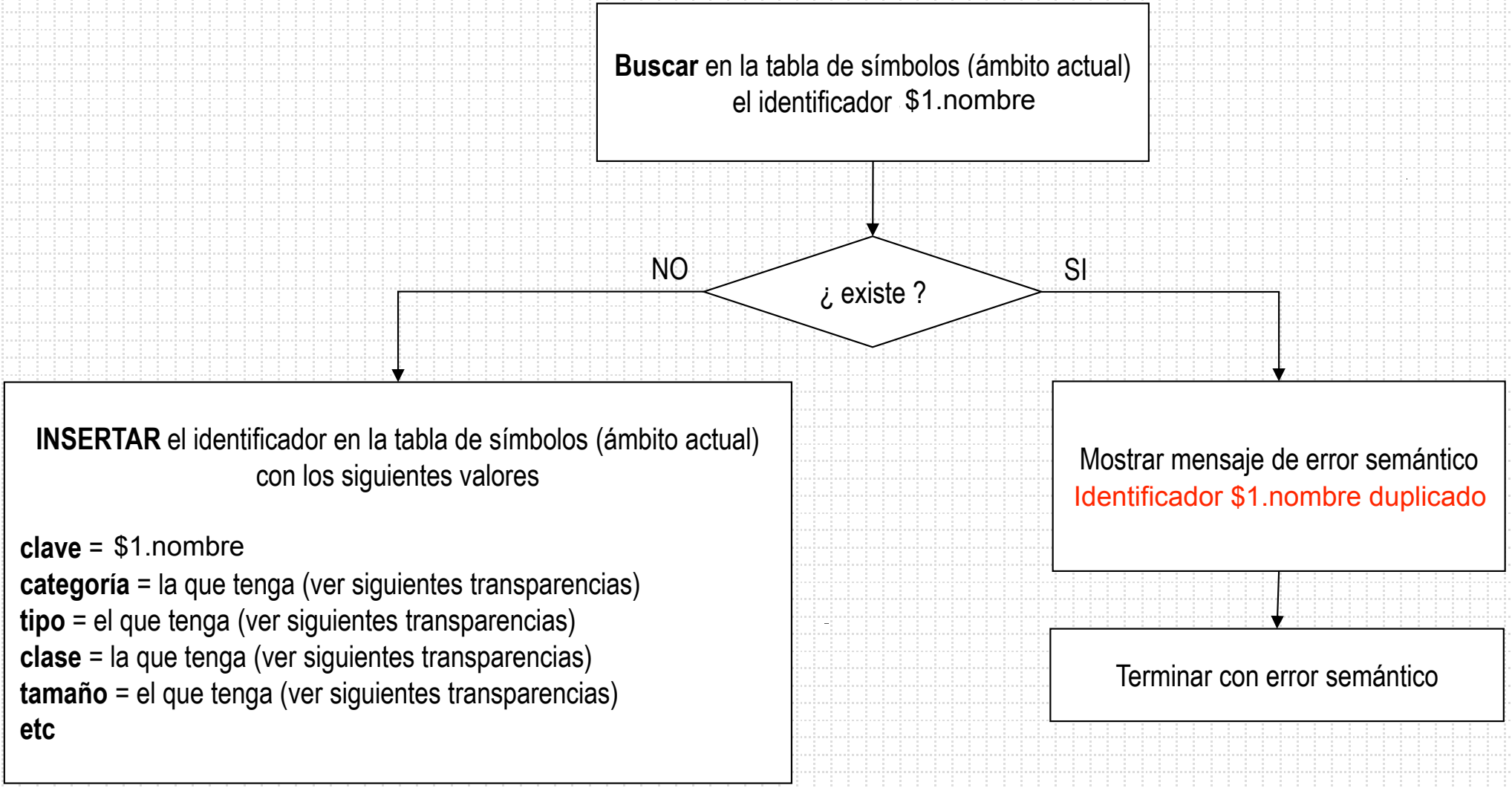
  identificador : TOK_IDENTIFICADOR

- El esquema general de la acción semántica se muestra en la siguiente figura.

- La búsqueda se realiza en el ámbito donde se efectúa la inserción, es decir, el ámbito actual.

Declaración de variables globales y locales

identificador : **TOK_IDENTIFICADOR**



Declaración de variables globales y locales

Propagación de las características de la declaración a las variables de la lista

- Una declaración de variables globales o locales en ALFA puede ser, por ejemplo:

```
int X, Y, Z;
```

en la que se declara que las variables X, Y y Z tienen las siguientes características:

- Son de tipo entero.
- Son de clase escalar.

- Otra declaración podría ser:

```
array boolean [8] A1, A2;
```

en la que se declara que las variables A1 y A2 tienen las siguientes características:

- Son de clase vector.
- El tipo base de los vectores es el tipo lógico.
- El tamaño de los vectores es 8.

Declaración de variables globales y locales

Propagación de las características de la declaración a las variables de la lista

- **Las características/propiedades** de una declaración de **variables globales o locales** son las siguientes:
  - **categoría** (variable)
  - **clase** (escalar o vector)
  - **tipo** (entero o lógico)
  - **número de elementos** (en el caso de vectores)
  - **posición** (en el caso de variable local)

Declaración de variables globales y locales

Propagación de las características de la declaración a las variables de la lista

- La producción correspondiente a una línea declarativa de variables globales o locales es la siguiente:

  (4) **declaracion: clase identificadores ';'**

  Como puede observarse, se realiza en primer lugar la reducción del no terminal **clase** y posteriormente la reducción del no terminal **identificadores**. Por lo tanto se puede deducir que:

  - Durante la reducción del no terminal clase se establecen las características de la declaración.
  - El no terminal identificadores tiene que "heredar" del no terminal clase las características de la declaración.

- Debido a que la herencia de atributos no es compatible con el análisis ascendente, es necesario establecer un mecanismo para implementar dicha herencia. El mecanismo más sencillo es el uso de variables globales, una para cada atributo heredado. A continuación se describen dichas variables.

Declaración de variables globales y locales

Propagación de las características de la declaración a las variables de la lista

- Una variable global, **tipo_actual**, para propagar el **tipo de la declaración**. Esta variable se actualiza con el valor correspondiente (entero o lógico) en las siguientes producciones:
  - (10) tipo : TOK_INT
  - (11) tipo : TOK_BOOLEAN
- Una variable global, **clase_actual**, para propagar la **clase de la declaración**. Esta variable se actualiza con el valor correspondiente (escalar o vector) en las siguientes producciones:
  - (5) clase : clase_escalar
  - (7) clase : clase_vector
- Una variable global, **tamanio_vector_actual** para uso exclusivo en declaraciones de vectores. Permite propagar el **tamaño** del mismo. Esta variable se actualiza en la siguiente producción:
  - (15) clase_vector : TOK_ARRAY tipo '[' constante_entera ']'

Declaración de variables globales y locales

Propagación de las características de la declaración a las variables de la lista

- Una variable global, **pos_variable_local_actual,** para propagar la **posición de una variable local** en declaraciones correspondientes a variables locales.
  Esta variable se puede gestionar de la siguiente manera:
  - se inicializa a 1 antes de la sección declarativa de una función.
  - se incrementa cada vez que se procese la declaración de una variable local, es decir, en la producción:

    identificador : TOK_IDENTIFICADOR

- En resumen, **el valor de las variables globales se establece en las acciones semánticas de las producciones adecuadas, según se ha explicado en los párrafos anteriores, con el objetivo de disponer de esos valores en las reducciones correspondientes a los identificadores que aparecen en la lista de variables de la declaración** (ver ejemplo en la transparencia 34)

Declaración de variables globales y locales

Acciones semánticas de las producciones 5 y 7

```
(5)     clase:    clase_escalar
                  {
                        clase_actual = ESCALAR[1];
                  }
(7)           |    clase_vector
                  {
                        clase_actual = VECTOR[1];
                  }
              ;
```

[1] Se asume que en "alfa.h" se define:

```
#define ESCALAR  1
#define VECTOR  2
```

Declaración de variables globales y locales

Acciones semánticas de las producciones 10 y 11

```
(10)    tipo:   TOK_INT
                {
                    tipo_actual = INT [1];
                }
(11)          | TOK_BOOLEAN
                {
                    tipo_actual = BOOLEAN [1];
                }
              ;
```

[1] Se asume que en "alfa.h" se define:

```
#define INT  1
#define BOOLEAN  2
```

Declaración de variables globales y locales

Acción semántica de la producción 15

- En la producción 15 se puede sustituir el no terminal constante_entera por el terminal TOK_CONSTANTE_ENTERA para evitar la reducción de la regla:

```
constante_entera : TOK_CONSTANTE_ENTERA
```

```
(15)    clase_vector:  TOK_ARRAY tipo '[' TOK_CONSTANTE_ENTERA ']'
                       {
                           tamanio_vector_actual = $4.valor_entero;
                           if  ((tamanio_vector_actual < 1 )  | |
                               (tamanio_vector_actual > MAX_TAMANIO_VECTOR [1]))
                            {
                               MOSTRAR MENSAJE ERROR SEMÁNTICO
                               TERMINAR CON ERROR
                            }
                       }
                       ;
```

[1] Se asume que en "alfa.h" se define:

```
#define MAX_TAMANIO_VECTOR  64
```

Declaración de variables globales y locales

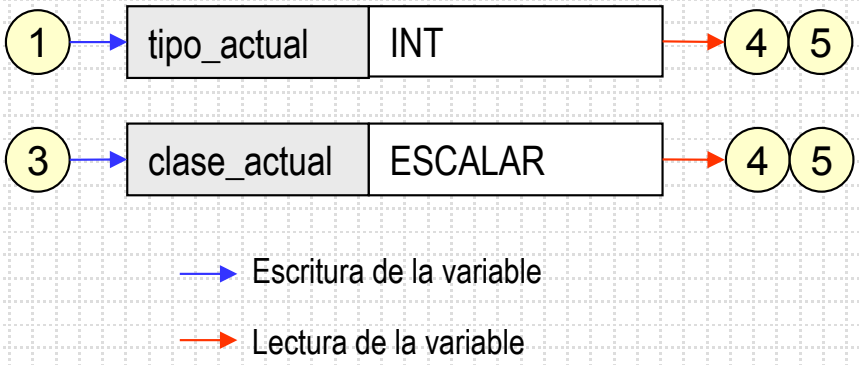
Ejemplo

- La declaración **int X, Y;** se analiza como se muestra en la figura

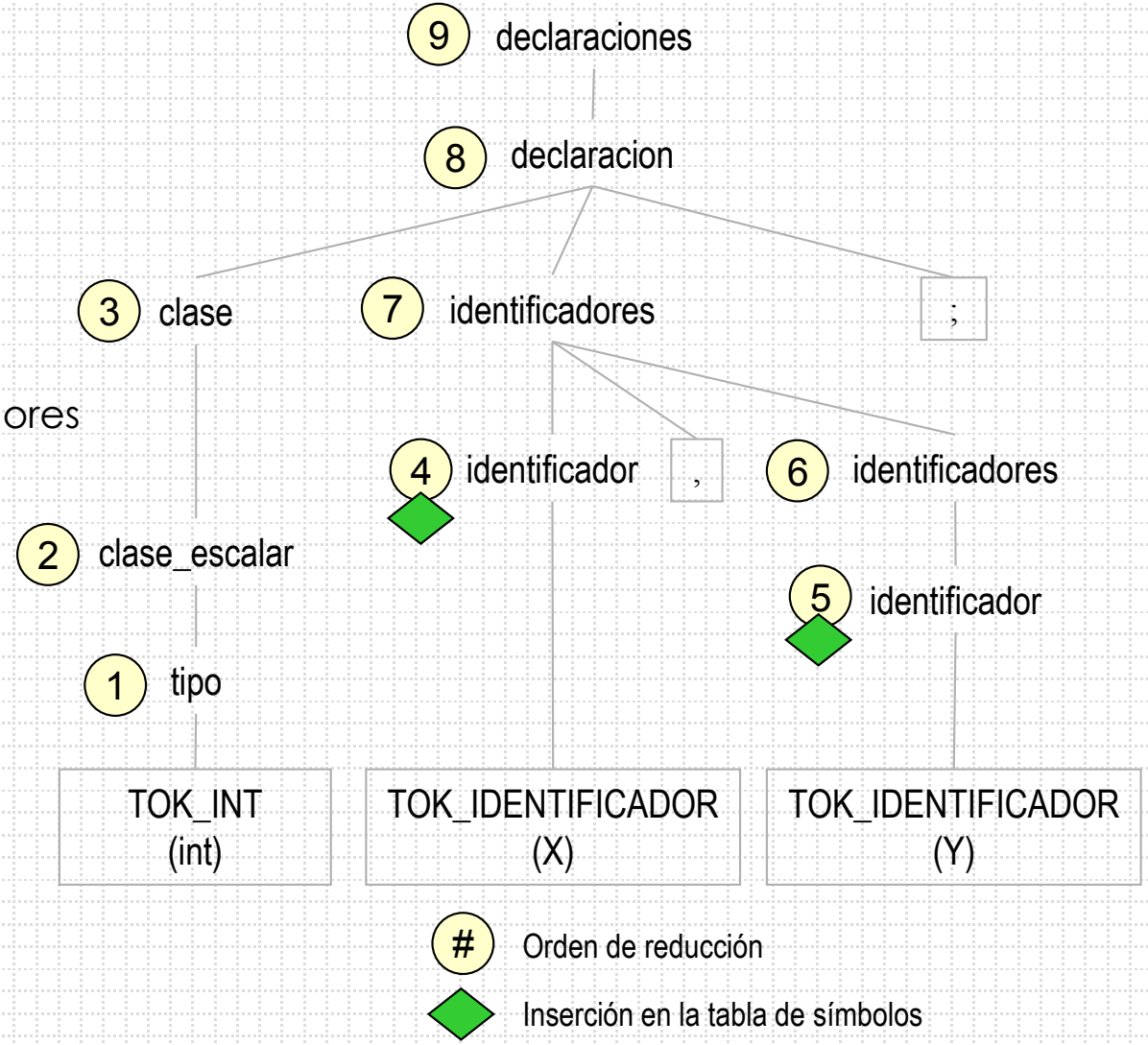
PRODUCCIONES IMPLICADAS

(2) declaraciones : declaracion
(4) declaracion : clase identificadores ';'
(5) clase : clase_escalar
(9) clase_escalar : tipo
(10) tipo : TOK_INT
(18) identificadores : identificador
(19) identificadores : identificador ',' identificadores
identificador : TOK_IDENTIFICADOR

VARIABLES GLOBALES



ÁRBOL DE ANÁLISIS



Declaración de variables globales y locales

Acción semántica adecuada al ámbito de la variable

- En la acción semántica de la regla del no terminal identificador, se requiere un mecanismo que permita saber si se está realizando la inserción de una variable local o una variable global ya que el tratamiento es ligeramente diferente.

- **Inserción de variable global:**
  - Se comprueba que el identificador no exista en el ámbito actual. Si existiera, se genera un mensaje de error semántico y se termina el proceso de compilación con error.
  - Si el identificador no existe en el ámbito actual, se realiza la inserción utilizando los valores almacenados en las variables globales (tipo_actual, clase_actual, etc)

- **Inserción de variable local:**
  - Se comprueba que el identificador no exista en el ámbito actual. Si existiera, se genera un mensaje de error semántico y se termina el proceso de compilación con error.
  - Se comprueba que la clase de la variable local sea escalar. Si no es así se genera un mensaje de error semántico y se termina el proceso de compilación con error.
  - Si no hay error semántico se realiza la inserción sin olvidar el uso de la variable global pos_variable_local_actual.
  - Se incrementa en uno el valor de las siguientes variables globales:
    - pos_variable_local_actual
    - num_variables_locales_actual (esta variable almacena el número de variables locales de una función)

      Se asume que estas variables son correctamente inicializadas (consultar documentación de la gestión de identificadores de funciones)

Declaración de parámetros de función

- Para procesar la inserción de los identificadores de parámetros se diseña una nueva producción específica de la siguiente manera:

  idpf : TOK_IDENTIFICADOR

- Es necesario modificar la producción 27 para mantener la coherencia con la nueva producción. La regla 27 quedaría de la siguiente manera:

  (27) parametro_funcion : tipo **idpf**

- La acción semántica de esta nueva producción se estructura así:
  - Se comprueba que el identificador no exista en el ámbito actual (el de la función). Si existiera, se genera un mensaje de error semántico y se termina el proceso de compilación con error.
  - Si no hay error semántico se realiza la inserción sin olvidar el uso de la variable global pos_parametro_actual.
  - Se incrementa en uno el valor de las siguientes variables globales:
    - pos_parametro_actual
    - num_parametros_actual (esta variable almacena el número de parámetros de una función)

    Se asume que estas variables son correctamente inicializadas (consultar documentación de la gestión de identificadores de funciones)

Declaración de funciones

- La inserción de los identificadores de funciones en la tabla de símbolos puede resolverse con el esquema de acciones que se marcan en la siguiente producción:

(22) funcion : TOK_FUNCTION tipo TOK_IDENTIFICADOR ❶
'(' parametros_funcion ')' '{' declaraciones_funcion ❷
sentencias '}' ❸

- La situación de las acciones en puntos intermedios de la producción sugiere la modificación de la gramática original de la siguiente manera:

fn_name : TOK_FUNCTION tipo TOK_IDENTIFICADOR ❶
fn_declaration : fn_name '(' parametros_funcion ')' '{' declaraciones_funcion ❷
funcion : fn_declaration sentencias '}' ❸

Declaración de funciones

- La primera producción añadida a la gramática es:

  fn_name : TOK_FUNCTION tipo TOK_IDENTIFICADOR ❶

- La acción del punto ❶ implica las siguientes tareas:
  - Buscar el identificador en el ámbito actual (el exterior a la función). Generar un error semántico si ya existe y terminar el proceso de compilación con error.
  - Si no ha habido error, abrir un ámbito nuevo. Esta operación implica:
    - insertar el identificador de la función en el ámbito actual (el exterior a la función)
    - crear un nuevo ámbito (el de la función) e insertar en él el identificador de la función.

    En este punto la información disponible acerca del identificador de la función se reduce al tipo de retorno de la misma. En los otros puntos marcados se completará esa información.
  - Inicializar las siguientes variables globales:
    - num_variables_locales_actual = 0
    - pos_variable_local_actual = 1
    - num_parametros_actual = 0
    - pos_parametro_actual = 0
  - Propagar a la parte izquierda de la regla el lexema del identificador de la función para su posterior uso en la acción del punto marcado como ❷.

Declaración de funciones

- La segunda producción añadida a la gramática es la siguiente:

  fn_declaration : fn_name ‘(‘ parametros_funcion ‘)’ ‘{‘ declaraciones_funcion ❷

- En la regla se puede observar que después de procesar el nombre de la función como consecuencia de la reducción del nuevo no terminal fn_name, se reducen los no terminales correspondientes a los parámetros de la función y a sus variables locales.

- La gestión de los identificadores de los parámetros y las variables locales se realiza como ya se ha expuesto anteriormente.

- En el punto ❷ ya se conoce el número de parámetros de la función, y es necesario actualizar en ese punto dicha información en la tabla de símbolos del ámbito actual para poder permitir el control semántico de dicha propiedad en las llamadas recursivas (si las hubiera)

Declaración de funciones

- La acción del punto ❷ implica las siguientes tareas:
  - Acceder al identificador de la función en el ámbito actual (el de la función) y actualizar la información correspondiente al número de parámetros.
    El acceso al identificador se puede realizar haciendo uso del lexema guardado como atributo del no terminal fn_name que se propagó en la acción ❶.
  - Propagar a la parte izquierda de la regla el lexema del identificador de la función para su posterior uso en la acción del punto marcado como ❸.

Declaración de funciones

- La inserción en la gramática de los dos nuevos símbolos fn_name y fn_declaration hace que la producción 22 de la gramática original quede modificada de la siguiente manera:

  funcion : fn_declaration sentencias '}' ❸

- La acción del punto ❸ implica las siguientes tareas:
  - Cerrar el ámbito de la función.
  - Acceder al identificador de la función en el ámbito actual (el exterior a la función) y actualizar la información correspondiente al número de parámetros.
    El acceso al identificador se puede realizar haciendo uso del lexema guardado como atributo del no terminal fn_declaration que se propagó en la acción ❷.

Uso de identificadores

- Cada vez que se **usa** un identificador hay que comprobar si se ha **declarado previamente**.
- Para comprobar si un identificador ha sido declarado, simplemente se **busca en la tabla de símbolos**. Si la búsqueda termina con existo significa que el identificador ha sido declarado y que en la compilación de su declaración se insertó en la tabla de símbolos. Es importante no olvidar que la **búsqueda se debe realizar en el ámbito actual y en todos los ámbitos que lo engloban**.
- Las producciones de la gramática que reflejan el **uso de identificadores** son aquellas pertenecientes a la **sección de sentencias** en las que aparece el no terminal "identificador".
- La producción del no terminal "identificador" ya está reservada para realizar la inserción de identificadores en la tabla de símbolos. Por lo tanto, es necesario **sustituir en las producciones de la gramática que reflejan el uso de identificadores la aparición del no terminal "identificador" por el terminal TOK_IDENTIFICADOR.**

Uso de identificadores

- Algunas de las producciones que se tienen que modificar son las siguientes:
  - (43) asignacion : TOK_IDENTIFICADOR '=' exp

  .....
  - (48) elemento_vector : TOK_IDENTIFICADOR '[' exp ']'

  .....
  - (54) lectura : TOK_SCANF TOK_IDENTIFICADOR

  .....
  - (80) exp : TOK_IDENTIFICADOR

  .....
  - etc

Uso de identificadores

- Ejemplo

```
lectura:   TOK_SCANF  TOK_IDENTIFICADOR
           {
                Buscar en la tabla de símbolos (todos los ámbitos abiertos) el identificador  $2.lexema
                Si no existe
                 {
                    MOSTRAR MENSAJE ERROR SEMÁNTICO
                    TERMINAR CON ERROR
                 }
                En caso contrario hacer lo correspondiente a la producción
                 {
                    COMPROBACIONES  SEMÁNTICAS
                        Si el identificador es una función   → ERROR SEMÁNTICO
                        Si el identificador es un vector  → ERROR SEMÁNTICO

                    GENERACIÓN DE CÓDIGO: APILAR LA DIRECCIÓN DEL IDENTIFICADOR
                        Si el identificador en una variable global , su dirección  es su lexema
                        Si el identificador es un parámetro o una variable local, su dirección se expresa en función de ebp y la
                        posición del  parámetro o variable local

                    GENERACIÓN DE CÓDIGO: LLAMAR A LA FUNCIÓN DE LA LIBRERÍA ALFALIB.O
                        Si el identificador es de tipo entero llamar a scan_int
                        Si el identificador es de tipo lógico llamar a scan_boolean
                        Restaurar la pila
                 }
           }
           ;
```