Procesadores de Lenguajes

Tema 3

El gestor de la tabla de símbolos

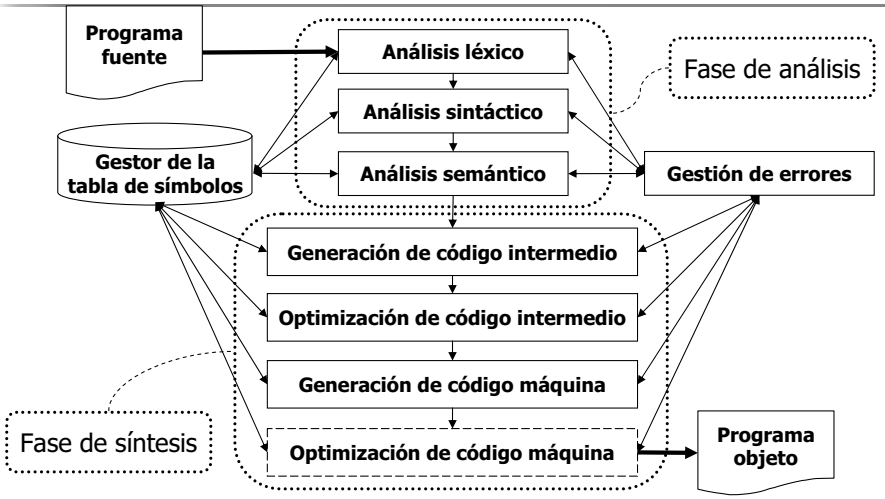
Tema 3 – El gestor de la tabla de símbolos

Sesión 1: Conceptos básicos

Sesión 1: Conceptos básicos

- El gestor de la tabla de símbolos (TS) en contexto
- Aspectos básicos del gestor de la TS
- Campos de una entrada de la TS (los atributos de un identificador)
- Tabla de símbolos y palabras clave
- Aspectos de diseño e implementación

El gestor de la tabla de símbolos en contexto



Gestor de la tabla de símbolos: aspectos básicos

- Definición (*tabla de símbolos – TS*):
  - Estructura de datos que contiene (al menos) un registro por cada identificador del programa fuente,
  - con campos para cada uno de los atributos del mismo.
- Funciones básicas del gestor de la TS:
  - Registrar los identificadores usados en el programa fuente.
  - Almacenar información sobre los distintos atributos de cada identificador.
- Requisitos básicos del gestor de la TS:
  - Búsqueda eficiente (rápida) del registro asociado a cada identificador.
  - Almacenamiento y consulta eficiente (rápido) de los campos del registro de un identificador (sus atributos).

Campos de una entrada de la TS

ENTRADAS HETEROGÉNEAS !!!

- Los campos de una entrada de la TS (los atributos de un identificador) son, **entre otros**:
  - Generales:
    - El lexema (clave)
    - El ámbito (bloque del programa donde es válido) (¿clave?)
    - El tipo semántico (INTEGER, PROC, FUN, etc.)
    - El espacio de memoria asignado
  - Nombres de subprogramas:
    - El número y el tipo de sus argumentos
    - El modo de paso de los argumentos (valor, referencia, etc.)
    - El tipo del retorno (VOID en el caso de los procedimientos)
  - Arrays:
    - Número de dimensiones
    - Número de componentes en cada dimensión

Análisis léxico

Análisis sintáctico

Análisis semántico

Generación de código intermedio

Tabla de símbolos y palabras clave

- Si las palabras clave son palabras reservadas:
  - Opción 1.1:
    - Se codifican en el traductor finito del escáner
  - Opción 1.2:
    - Se almacenan al arrancar el compilador dentro de la TS:
      - Mediante un campo que indica que la entrada es una palabra reservada (TipoToken)
      - En una tabla adjunta sólo para palabras reservadas
- En caso contrario:
  - A veces, un **lexema** del programa fuente es ambiguo: puede ser un identificador o una palabra clave (en función del contexto):
    - Opción 2: Se incluye el lexema en la TS con un AVISO de "posible conflicto léxico-sintáctico-semántico".

Aspectos de diseño e implementación (1)

- Gestión de los lexemas (a):

Con un tamaño máximo para la longitud del lexema

LEXEMA	RESTO de ATRIBUTOS
...	...
num	...
...	...
EscribeNumero	...

- Gestión de los lexemas (b):

Buffer (global) de lexemas

LEXEMA	RESTO de ATRIBUTOS
...	...
	...
...	...
	...

...	EOS	n	u	m	EOS	E	s	c	r	i	b	e	N	u	m	e	r	o	EOS	...

Aspectos de diseño e implementación (2)

- Estructura lineal:
  - En memoria estática.
    - Se almacena el lexema en el siguiente hueco libre.
  - En memoria dinámica.
    - En orden de aparición o declaración de identificadores (nombres).
    - Ordenada por lexemas.
- Tabla *hash* (con todas sus alternativas)
- Árbol de decisión:
  - Con tantos niveles como el máximo número de caracteres de los lexemas insertados.

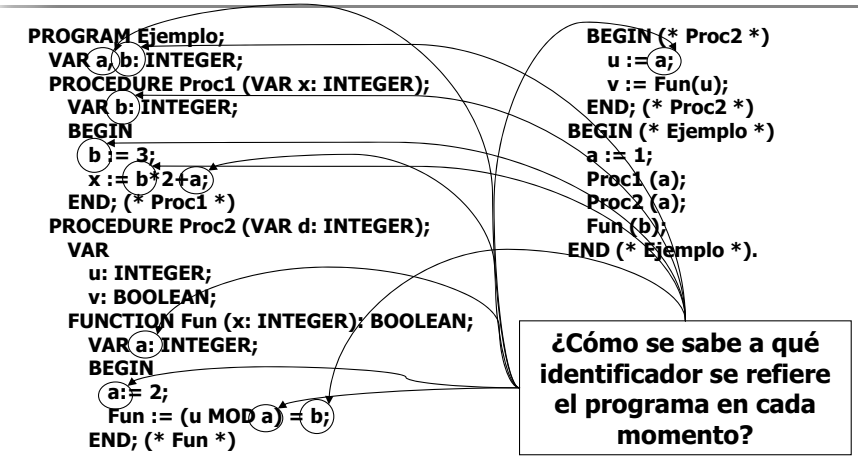
Tema 3 – El gestor de la tabla de símbolos

Sesión 2: Gestión de ámbitos

Gestión de ámbitos

- Gestión de ámbitos
  - Generalidades
  - Alternativas
  - Ejercicio 1: con anidamiento
  - Ejercicio 2: sin anidamiento
- Interfaz genérico del gestor de la TS

Ejemplo introductorio



Gestión de ámbitos – generalidades

- Se mantiene una tabla de símbolos (TS) separada para cada ámbito.
  - Las variables locales se almacenan dentro de una TS especialmente creada para el módulo, subprograma, bloque, etc.
- El analizador sintáctico/semántico (el traductor dirigido por la sintaxis) se encarga de la apertura y cierre de cada nueva TS.
- Algoritmo de búsqueda de lexemas:
  - Caso básico:
    - Se busca el *lexema* en la TS del ámbito actual, A_i (TS activa).
  - Caso recursivo:
    - Si no se ha encontrado el *lexema* en la TS de A_i:
      - Si A_i está contenido en un ámbito más global (A_{i-1}), se busca el *lexema* en la TS de A_{i-1}.
      - En caso contrario, el *lexema* no está en la tabla de símbolos.

Gestión de ámbitos – alternativas (1)

- Hay que tener identificada la TS de ámbito superior que contiene a la actualmente activa.
  - Solución 1:
    - El anidamiento se gestiona mediante una pila.
  - Solución 2:
    - Se mantienen dos atributos, continente y contenido, dentro de la TS asociada a cada ámbito:
      - **Continente:** puntero de una TS a aquella otra que almacena la información de un ámbito superior, en el que se contiene el que gestiona la TS actual.
      - **Contenido:** lista de punteros de una TS a aquellas otras que almacenan la información de un ámbito inferior, contenido en el que gestiona la TS en cuestión.

Gestión de ámbitos – alternativas (2)

- Solución 3:
  - Se numera cada nuevo ámbito distinto.
    - Cuidado: los registros o los bloques de C, por ejemplo, tienen un ámbito propio.
  - Se mantiene una estructura auxiliar, la **matriz de bloques**, que determina qué bloque es representado en cada TS:

NÚMERO de BLOQUE	PUNTERO a la TS de NÚMERO de BLOQUE	NÚMERO de BLOQUE del CONTINENTE (opcional)
…	…	…

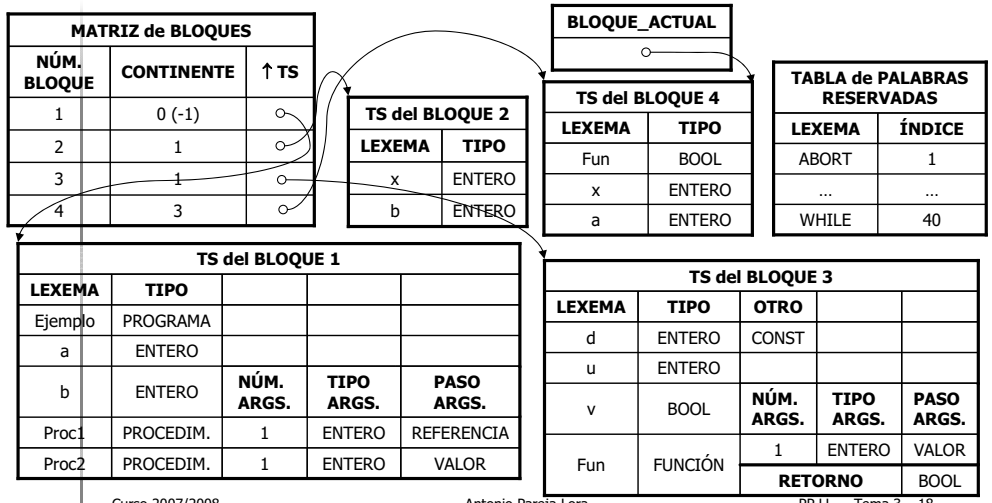
Gestión de ámbitos – alternativas (3)

- Solución 3 (cont.):
  - La clave de la TS ahora está compuesta por:
    - El lexema del identificador
    - El número (o identificador del ámbito) al que pertenece el lexema (aquél en el que se declara) (atributo **ámbito**)
- Tanto en la solución (2) como en la (3):
  - Hay que introducir una nueva variable en el compilador para saber cuál es la TS activa en cada momento (**bloque_actual**).

Ejemplo introductorio (bloques)

```
PROGRAM Ejemplo;
  VAR a, b: INTEGER;
  PROCEDURE Proc1 (VAR x: INTEGER);
    VAR b: INTEGER;
    BEGIN                          BLOQUE 2
      b := 3;
      x := b*2+a;
    END; (* Proc1 *)
  PROCEDURE Proc2 (d: INTEGER);
    VAR
      u: INTEGER;                  BLOQUE 3
      v: BOOLEAN;
    FUNCTION Fun (x: INTEGER): BOOLEAN;
      VAR a: INTEGER;
      BEGIN                        BLOQUE 4
        a:= 2;
        Fun := (u MOD a) = b;
      END; (* Fun *)
    BEGIN (* Proc2 *)
      u := a;
      v := Fun(u);
    END; (* Proc2 *)
BLOQUE 3 (cont.)
BEGIN (* Ejemplo *)
  a := 1;
  Proc1 (a);
  Proc2 (a);
  Fun (b);
END (* Ejemplo *).
                                   BLOQUE 1
```

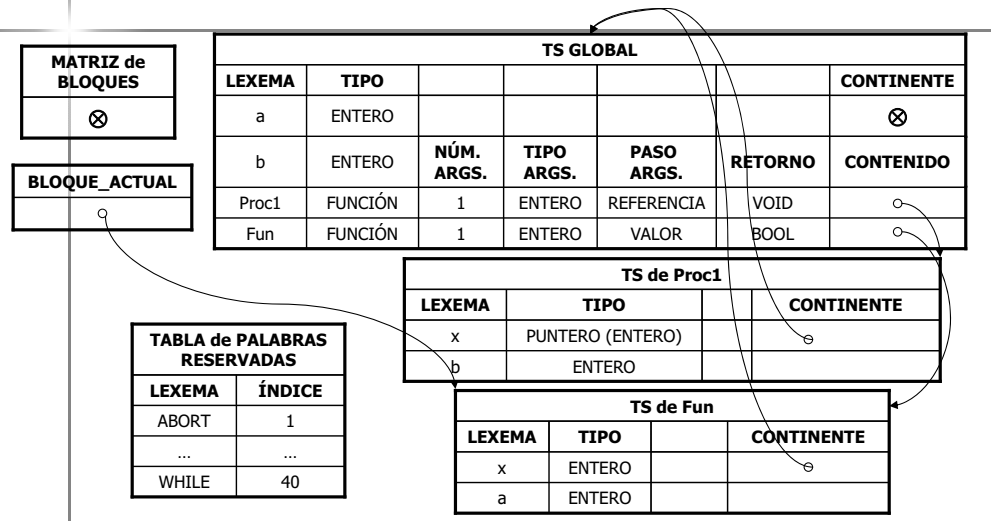

Ejemplo introductorio (traza)



Ejemplo 2 (bloques)

```
int a, b;                          BLOQUE 1

void Proc1 (int *x)
{  int b;
   b = 3;                          BLOQUE 2
   *x = b*2+a;
}

boolean Fun (int x)
{  int a;
   a = 2;                          BLOQUE 3
   return (u % a) == b;
}

void Proc2 (int d)
{  int u;
   boolean v;
   u = a;                          BLOQUE 4
   v = Fun (u);
}

void main ()
{  a = 1;
   Proc1 (&a);
   Proc2 (a);
   Fun (b);
}
```


Ejemplo 2 (traza)



Interfaz genérico del gestor de la TS (1)

- Con el analizador léxico:
  - Teniendo en cuenta que:
    - TipoToken ≡ (TipoCodToken × TipoAtributo)
      - TipoCodToken ≡ (ID, PAL_RES, OP, PUNT, …)
      - TipoAtributo ≡ PUNTERO | ENTERO | REAL | ∅ | …
  - ↑ $ENTRADA_{TS}$ Inserta (STRING, TipoCodToken);
  - ↑ $ENTRADA_{TS}$ Busca (STRING);
- Con el resto de módulos, en general:
  - Teniendo en cuenta que:
    - TipoCampo ≡ (LEXEMA, TIPO_SEM, NUM_ARGS, TIPO_ARGS, …)
  - ↑$ENTRADA_{TS}$ Completa (↑$ENTRADA_{TS}$,TipoCampo,TipoValor);
  - TipoValor Consulta (↑$ENTRADA_{TS}$, TipoCampo);

Interfaz genérico del gestor de la TS (2)

- Con el analizador sintáctico y el semántico:
  - ↑ TS CreaTabla (↑ TS);
    - Se pasa un puntero a la tabla activa, que se convierte en la del ámbito continente de aquél recogido en la nueva TS que se crea.
    - Se retorna un puntero a la nueva tabla activa, que acaba de crearse, para un ámbito más interno.
  - ↑ TS Desactiva (↑ TS);
    - La tabla que se pasa como parámetro deja de ser la tabla activa, pasando a serlo la tabla del ámbito que contiene al que la primera representa.
    - Se retorna un puntero a la nueva tabla activa, que es la del continente del que se desactiva.
    - La tabla no se destruye si se desea guardar información para depuración de programas.

Bibliografía

- Aho, A. V.; Sethi, R.; Ullman, J. D.: *Compilers: Principles, Techniques and Tools*. Massachusetts: Addison-Wesley Publishing Company, 1986.
- Alfonseca Cubero, E.; Alfonseca Moreno, M.; Moriyón Salomón, R. Teoría de autómatas y lenguajes formales. Madrid: Mc-Graw-Hill/Interamericana de España, S.A.U., 2007.
- Grogono, P. Programación en Pascal. Wilmington, Delaware (EE.UU.):Addison-Wesley Iberoamericana, 1996.
- Sanchís Llorca, F. J. y Galán Pascual, C. Compiladores: Teoría y construcción. Madrid: Editorial Paraninfo, 1986.