LA TABLA DE SIMBOLOS

1. Estructura de la tabla de símbolos

La tabla de símbolos es la estructura de datos usada por el compilador para asociar a cada símbolo del programa fuente (identificadores, constantes, etc...) un contenido semántico. Esta formada por registros que en general tienen una longitud fija. Los campos que tienen estos registros varían dependiendo del lenguaje a compilar y de la estructura del compilador. En general, estos campos pueden contener la información misma para la que estan destinados, o bien un puntero a posiciones de memoria en las que se guarda esta información. Algunos de los campos que pueden definirse para los registros de la tabla de símbolos son:

? Nombre del símbolo. En la tabla de símbolo deben aparecer todos los lexemas correspondientes a los símbolos usados en el texto fuente, los símbolos serán todos aquellos trozos de código que representen identificadores, constantes númericas, o cualquier otro elemento que el lenguaje sea capaz de manejar. Para el almacenamiento en la tabla de símbolos de los lexemas pueden utilizarse las técnicas:

  Interna. El campo de la tabla de símbolos se define como de tipo cadena de caracteres de dimensión fija (por ejemplo 16). El inconveniente de esta técnica es que la longitud máxima de los identificadores esta restringida al valor fijado
  Externa. El campo de la tabla de símbolos contiene un puntero a una zona de memoria en la que se almacena el lexema.

? Dirección en memória. En este campo se almacenan las direcciones de memoria en la que se guardarán los valores de las variables correspondientes a cada símbolo, durante la ejecución del programa. Normalmente el dato que interesa almacenar aquí no es la dirección absoluta, sino la relativa a una determinada zona de memoria de las que el programa usará en su ejecución. Por ejemplo, la dirección relativa al comienzo de la memória estatica, la direccion relativa al comienzo del registro de activación cotrrespondiente al procedimiento, etc...

  Es importante recordar que la tabla de símbolos no contiene los valores de las variables, sino simplemente la dirección en la que se almacenarán cuando el programa objeto se ejecute. Esto es obvio, ya que la tabla de símbolos es una estructura de datos que se utiliza en la fase de compilación, y por lo general, desaparece una vez terminada la compilación, no incorporandose al programa objeto. Por otra parte, sólo durante la ejecución del programa es cuando se computan los valores de las variables.

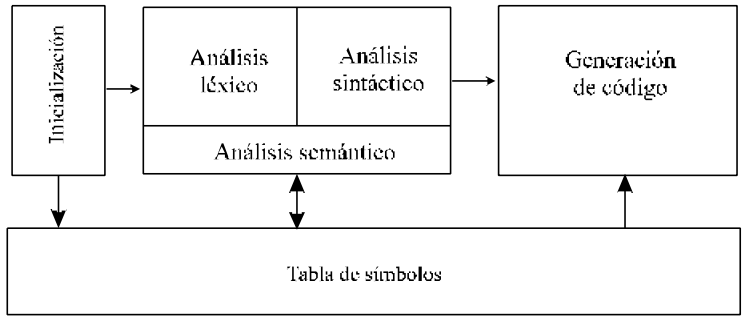
  Excepcionalmente, cuando queremos disponer de un depurador simbólico durante la fase de ejecución, no tenemos mas remedio que incorporar la tabla de símbolos al programa objeto. Esto ocurre también en los interpretes, ya que compilación y ejecución se alternan.

? Tipos. En este campo se almacenará el código correspondiente al tipo de datos representado por el símbolo, o bien un puntero a la estructura de datos correspondiente, según se ve en el capitulo dedicado a los sistemas de tipos.

? Número de linea de la declaracíon o números de linea en los que se usa el símbolo. Esta es una información auxiliar que será de utilidad para producir un listado de referencias cruzadas que ayuden a la depuración de los programas. Puede utilizarse una lista encadenada de números de lineas, siendo la cabeza de esta lista la linea en donde aparece la declaración del símbolo.

2. Funcionamiento de la tabla de símbolos dentro del compilador

La función de la tabla de símbolos dentro del compilador y su interacción con los distintos módulos que lo forman, puede represenarse mediante el siguiente esquema:



La tabla de símbolos puede inicializarse con cierta información sobre símbolos especiales en el lenguaje, como son las palabras reservadas, las funciones de librería, las constantes predefinidas, etc...

El analizador lexicográfico va leyendo el texto fuente y construyendo lexemas, creando nuevas entradas en la tabla conforme va encontrando nuevos símbolos, para lo cual evidentemente ha de comprobar que el símbolo no figure ya en la tabla. El analizador sintáctico trabaja solamente con la estructura formal subyacente, y a nivel de tokens ya formados en la fase anterior. El analizador sintáctico sirve de soporte o armazón de un conjunto de rutinas de anásis semántico. Parte de estas rutinas pueden completar la información de la tabla de símbolos sobre el tipo de un determinado lexema, etc.

El generador de código usa la información de la tabla para referenciar el código a las posiciones adecuadas de memória. Las siguientes fases del compilador y del montador, en general, no necesitan usar la tabla de símbolos.

Una vez más conviene recordar que es el compilador el que usa la tabla de símbolos y no el programa compilado. La tabla de símbolos no está presente durante la ejecución del programa (salvo en el caso de que se incluya un depurador simbólico o en los interpretes).

3. Operaciones con la tabla de símbolos

La tabla de símbolos funciona como una estructura de base de datos en la que el campo clave es el lexema correspondiente al símbolo. Sobre la estructura de datos "Tabla de símbolos" pueden hacerse por lo general las siguientes operaciones :

? Insertar: un registro nuevo y comprobar que no existe otro con el mismo nombre.

- Buscar: Localizar el contenido de la tabla de símbolos asociado a un determinado lexema.
- Modificar: la información contenida en un registro. En general, sólo se añade información. La operación de modificar siempre conlleva una operación de búsqueda previa.

En lenguajes con estructura de bloque se incluyen dos operaciones más :

- Nuevo_Bloque: Inicio de un Nuevo Bloque.
- Fin_Bloque: Terminación del ámbito de un bloque.

Las inserciones se realizan cuando se procesa la zona de declaración de variables. En general, las operaciones más frecuentes son las de consulta, que se realizan antes de cada inserción (para ver si esta declarado el símbolo), y cada vez que aparece el símbolo en el programa fuente. En las tablas de símbolos no suelen borrarse registros durante su funcionamiento, salvo en los lenguajes con estructura de bloques, en los que al finalizar el análisis de un bloque deben desalojarse de la tabla todos los símbolos locales al mismo.

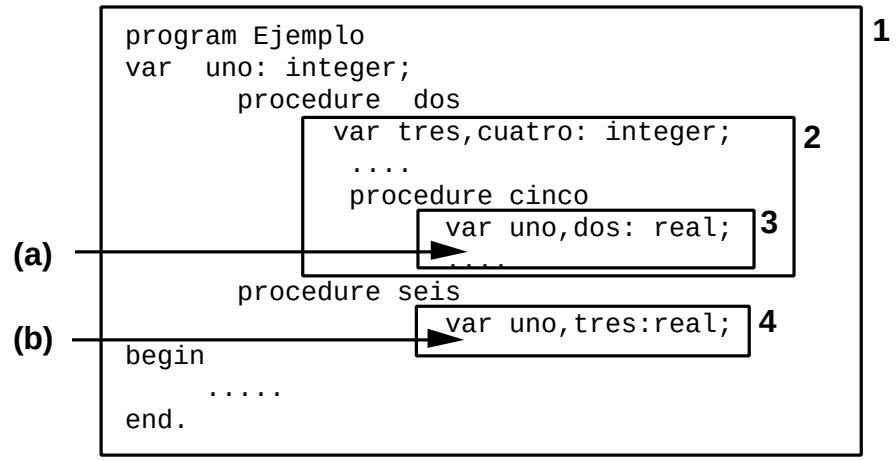
4. Organización de la tabla de símbolos

La organización de la tabla de símbolos es un problema convencional de programación, y por tanto no es tema especifico de esta materia. Algunos de los métodos que pueden emplearse son:

- Tablas no ordenadas. (Arrays, Listas, etc.) En general, son métodos muy poco eficientes, pero fáciles de programar.
- Tablas ordenadas. (Arrays ordenados, Listas ordenadas, Árboles binarios, Árboles AVL, Tablas “Hash”, etc.). Debido a las características específicas de funcionamiento de las tablas de símbolos, los métodos más eficientes son los árboles AVL y las tablas “Hash”

Sin embargo, el manejo de las tablas de símbolos en los lenguajes con estructura de bloques tiene unas características propias, y sugieren el empleo de técnicas especificas de organización para hacer más eficaces las operaciones de búsqueda y desalojo de registros una vez finalizada la compilación de un bloque.

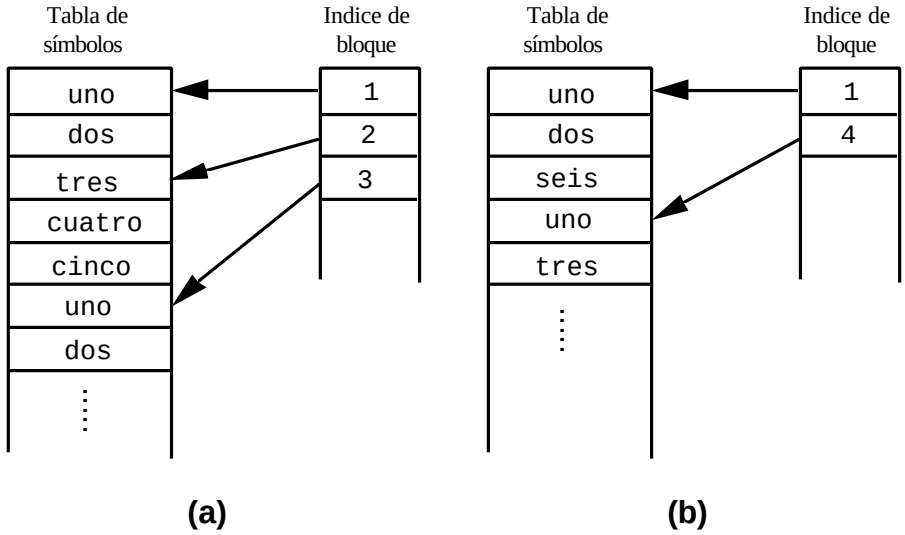
Un lenguaje con estructura de bloques es aquel que está compuesto por módulos o trozos de código cuya ejecución es secuencial. Estos módulos a su vez pueden contener en su secuencia de instrucciónes llamadas a otros módulos, que a su vez pueden llamar a otros submódulos y así sucesivamente. En general, un símbolo declarado en un módulo es accesible dentro del módulo en el que está definido y en todos los módulos o submodulos que contiene. Si uno de estos submoulos define un símbolo con el mismo lexema, esta definición se superpone a la existente anteriormente, asociando el lexema a la definición más reciente. Al conjunto de reglas semánticas que permiten deducir que definición corresponde a cada uno de los lexemas se denomina “*reglas de ámbito*”, y son una característica propia de cada lenguaje. La siguiente figura muestra la estructura de bloques de un programa PASCAL:



Las técnicas para el manejo de tablas de símbolos en lenguajes con estructura de bloques suelen emplear una pila, auxiliar de "Indices de Bloque" cuyos elementos, son punteros a la tabla de símbolos y sirven para marcar el comienzo (o el final) de los símbolos correspondientes a un bloque. Este mecanismo varía según la estructura de la tabla de símbolos:

(1) Tablas de símbolos no ordenadas. Tanto si se implementa con un *array* o con una lista, es el caso más sencillo. Cada vez que comienza un nuevo bloque se añade un puntero a la pila de "Indice de Bloque", que señala al primer símbolo del bloque. Cuando se termina de compilar el bloque se borran todos los símbolos que hay desde el que indica el último indice de bloque hasta el final de la tabla.

Por ejemplo, en el programa PASCAL de la siguiente figura anterior aparecen señalados cuatro bloques. Durante la compilación del programa, la configuración de la tabla de símbolos al alcanzar las marcas señaladas en el programa fuente se muestra en los siguientes esquemas:



Las operaciones sobre esta tabla son sencillas:

? Insetar: Previamente a la inserción de un elemento en la tabla se debe comprobar que no está ya definido en el mismo bloque. Esto es fácil ya que sólo requiere que el ámbito de la búsqueda se restrinja a los símbolos del último bloque, es decir, al

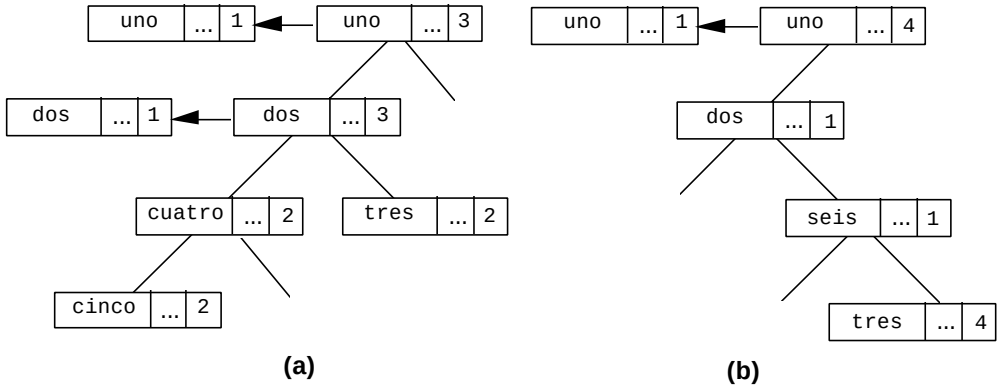
conjunto de registros que va desde el registro al que apunta el último indice de bloque hasta el final de la lista.

? Buscar: La búsqueda de un registro en la tabla debe realizarse en sentido inverso al de la inserción (de abajo hacia arriba), de manera que el primer encuentro con el lexema corresponde a la definición realizada en el bloque más próximo.

? Nuevo_Bloque: Basta con colocar un uevo elemento en la pila de "Indice de Bloque" que señale a la cima actual de la tabla de símbolos.

? Fin_Bloque: Se eliminan todos los registros contenidos en la tabla de símbolos desde el registro al que apunta el último indice de bloque hasta el final de la lista. Si es necesario al final de la compilación realizar un listado de todas las variables que aparecen en el programa, en vez de borrarlas se copian a otra estructura de datos independiente.

(2a) Tablas de símbolos con estructura de árbol: Árbol único. En los casos en que la tabla de símbolos se estructura como un árbol (ya sea un simple árbol binario, un árbol AVL, etc.), una solución al tratamiento de lenguajes con estructura de bloques consiste en añadir a los registros de la tabla un nuevo campo que indique el número del bloque al que pertenece el símbolo. En los nodos del árbol se utiliza una lista encadenada para solucionar las posibles colisiones de lexemas iguales en varios bloques. Siguiendo con el ejemplo anterior se obtendrían los siguientes esquemas:



Las operaciones sobre esta tabla son las siguientes:

? Insetar: Para insertar un nuevo símbolo en esta tabla se realiza una búsqueda previa de la posición que le corresponde. En caso de que exista un registro con la misma clave (el mismo lexema) se comprueba que el campo de bloque no corresponda al bloque activo, y se crea, partiendo de este nodo del árbol, una lista cuya cabeza corresponde al nuevo registro. En caso de que no exista ningún registro con la misma clave, se inserta normalmente.

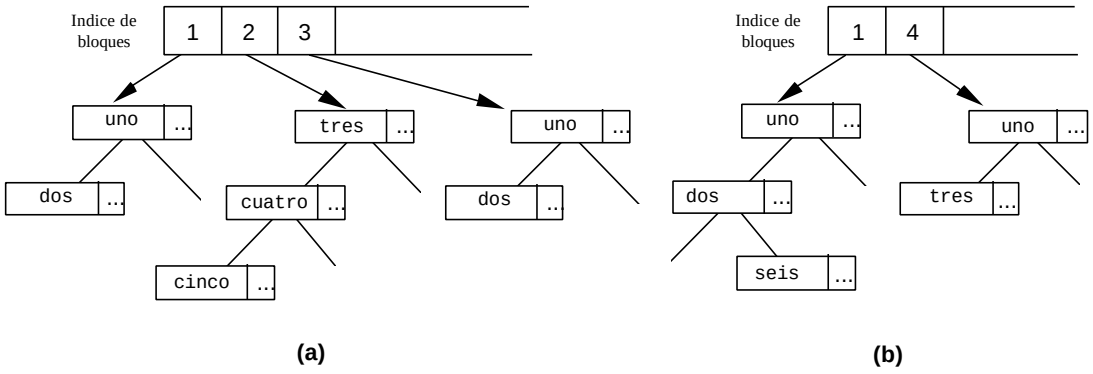
? Buscar: La búsqueda de un registro en la tabla se realiza como en cualquier árbol binario. Nunca es necesario recorrer las listas que puedan aparecer en cada nodo, ya que la estrategia de inserción garantiza que la definición más reciente es la que ocupa la cabeza de las listas y coniguientemente el nodo del árbol binario.

? Nuevo_Bloque: No hay que hacer nada más que registrar el cambio del bloque activo.

? Fin_Bloque: Este es el principal inconveniente de este método, ya que al finalizar el bloque es necesario eliminar los símbolos locales al bloque, lo que implica: (1) Localizar los registros del árbol correspondientes al bloque activo; y (2) borrarlas del árbol. Esta operación requiere un mayor exfuerzo computacional, especialmente

en árboles AVL, ya que la operación de borrado modifica la estructura del árbol y hace que deje de estar balanceado.

(2b) Tablas de símbolos con estructura de árbol: Bosque de árboles. En este caso la tabla de símbolos está formada por múltiples árboles, uno por cada bloque. Se usa una estructura de datos auxiliar de tipo pila que sirve de "Indice de bloques" Siguiendo con el ejemplo anterior se obtendrían los siguientes esquemas:

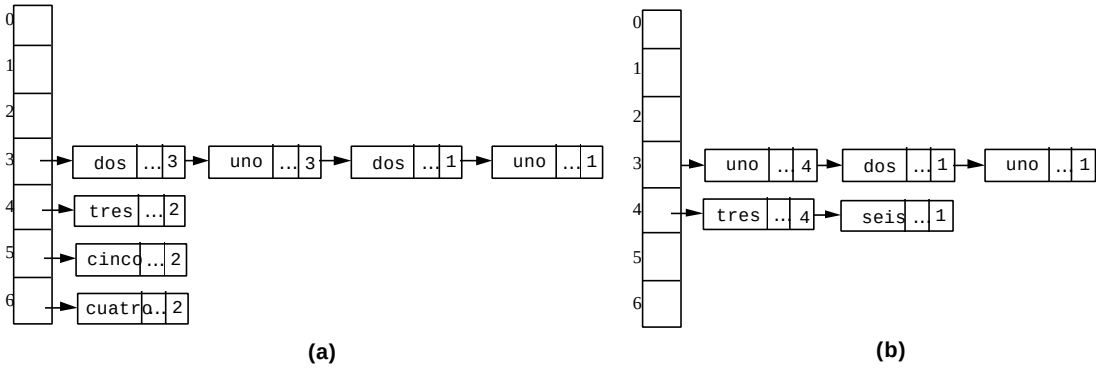


Las operaciones sobre esta tabla son las siguientes:

? Insetar: Al existir tablas independientes para cada bloque la operación de inserción se realiza normalmente, ateniendo sólo al bloque activo.
? Buscar: La búsqueda de un registro en la tabla es algo mas compleja que el caso anterior desde el punto de vista computacional. Para localizar un lexema se busca primero en la tabla del bloque activo, luego en la tabla a la que apunta el índice de bloque anterior, y así sucesivamente hasta localizarlo o hasta que se hayan rastreado todos los árboles.
? Nuevo_Bloque: Se crea un nuevo elemento en la pila de índices de bloques y una nueva estructura de tipo árbol inicialmente vacía.
? Fin_Bloque: Simplemente se destruye toda la estructura arborescente asociada al bloque que se termina, y se elimina el puntero de la pila de índices de bloques.

(3a) Tablas de símbolos "Hash": Tabla única. Al igual que con la estructura de tipo árbol, se puede emplear una única tabla añadiendo a los registros de la tabla un nuevo campo que indique el número del bloque al que pertenece el símbolo. En este caso el mecanismo que utiliza la técnica "Hash" para resolver las colisiones basta para resolver el problema de la multiplicidad de usos de un mismo lexema. Una de estas soluciones al problema de las colisiones es, al igual que en el caso anterior, utilizar una zona de desbordamientoque puede implementarse mediante una lista.

En el ejemplo anterior considerando una tabla "Hash" con desbordamiento se obtendrían los siguientes esquemas:



Las operaciones sobre esta tabla son las siguientes:

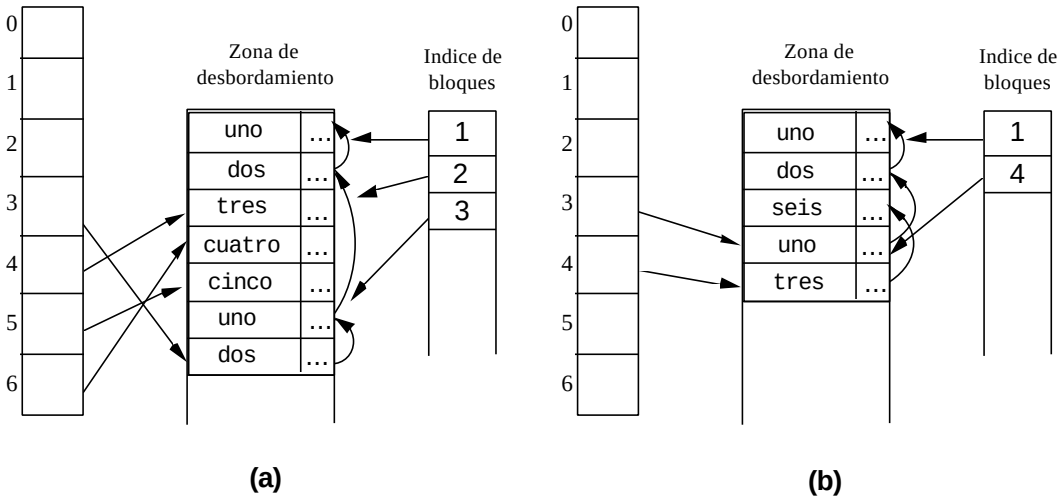
? Insetar: La inserción se realiza normálmente, como en cualquier tabla “Hash”. El tratamiento de las colisiones más indicado es el uso de listas de desbordamiento, aunque puede emplearse cualquier otro sistema con tal de que incluya los elementos más recientes en las primeras posiciones.

? Buscar: Como en cualquier tabla “Hash”, en caso de colisión de la clave que devuelve la función “hash”, debe buscarse en la zona de desbordamiento, hasta enconrar el símbolo, o hasta encontrar una posición vacía.

? Nuevo_Bloque: No hay que hacer nada más que registrar el cambio del bloque activo.

? Fin_Bloque: Como en el caso de los árboles únicos, éste es el principal inconveniente de este método, ya que al finalizar el bloque es necesario eliminar los símbolos locales al bloque, lo que implica: (1) Localizar todos los registros de la tabla “Hash” correspondientes al bloque activo; y (2) borrarlas de la tabla. Ésto presenta problemas desde el punto de vista computacional, ya que para localizar los símbolos de un bloque hay que recorrer prácticamente la tabla completa. Además, el borrado de elementos en una tabla “Hash” no es precisamente una tarea sencilla si las colisiones se resuelven sobre la propia tabla.

A fin de solucionar este problema, y dadas las características especificas de funcionamiento de la tabla de símbolos, puede modificarse el método utilizando una zona de memoria contigua para la zona de desbordaiento, y al igual que en otras ocasiones, una estructura auxiliar de tipo pila para los indices de bloques.Siguiendo con el ejemplo anterior se obtendrían los siguientes esquemas:



Sobre esta tabla con una zona de desbordamiento contigua, las operaciones de insertar y buscar son las mismas que en el caso anterior, las operaciones de bloques se modifican de la siguiente forma:

? Nuevo_Bloque: Se crea un nuevo puntero que se sitúa en la cima de la pila de indices de bloques y que señala al final de la zona de desbordamiento.

? Fin_Bloque: Dado que la zona de desbordamiento es contigua, la operación de borrado de los registros de un bloque es muy fácil, basta con eliminar todos los registros desde el final de la zona de desbordamiento hasta la posición que apunta el índice de bloques anterior. El borrado de los símbolos debe hacerse de uno en uno en orden inverso a la entrada, para que pueda mantenerse el puntero inicial de la tabla “Hash” que señala a la cabecera de la correspondiente lista de desbordamiento.