Compiladores
Facultad de Ciencias UNAM
Tabla de Símbolos

Lourdes Del Carmen González Huesca [*]

15 de febrero de 2024

Las tablas de símbolos son estructuras de datos creadas y utilizadas en tiempo de compilación. Esta tabla entre otras cosas, se encarga de almacenar información sobre los tokens del programa fuente, vitales para la compilación.

Las fases del front-end que son las encargadas de analizar el programa fuente, recolectan información de forma *incremental* en la tabla de símbolos. Durante estas fases la tabla relaciona nombres léxicos y sus atributos. Esta información será utilizada por las fases de back-end, encargadas de la síntesis del código para generar finalmente el código objeto.

Estas tablas deben de estar diseñadas para que el compilador pueda almacenar y recuperar de forma rápida y eficiente la información sobre identificadores, variables, declaraciones, nombres de procedimientos o funciones, etc.

La tabla de símbolos tiene rubros para almacenar información sobre cada identificador del programa tales como su lexema, su tipo, la posición de almacenamiento (dirección relativa) y alguna otra información relevante, como por ejemplo el tipo de dato, palabras reservadas, operadores e identificadores.

En términos de diseño de la tabla, se debe permitir que la tabla pueda almacenar múltiples declaraciones del mismo identificador, ya que puede aparecer en diferentes posiciones o repetirse en diferentes contextos. Debido a esto es necesario establecer el alcance de cada identificador.

Recordemos que el **alcance** se define como la porción de programa a la cual una declaracón se aplica y generalmente se define de manera estática, es decir en tiempo de compilación. Dependiendo del alcance de cada identificador, se generará una tabla de símbolos diferente para no tener conflictos y para asegurar que la información que se almacene en cada tabla sea consistente con el programa fuente. De esta forma, cada unidad de un programa como un bloque, función o método, que tiene sus propias declaraciones tendrá su propia tabla de símbolos con rubros para cada declaración en ella. Esta metodología se extiende a otras unidades de programas como clases, secciones, módulos, etc.

Las entradas o rubros en una tabla de símbolos son creadas y usadas en la etapa de análisis tanto por el analizador léxico, el sintáctico y el semántico:

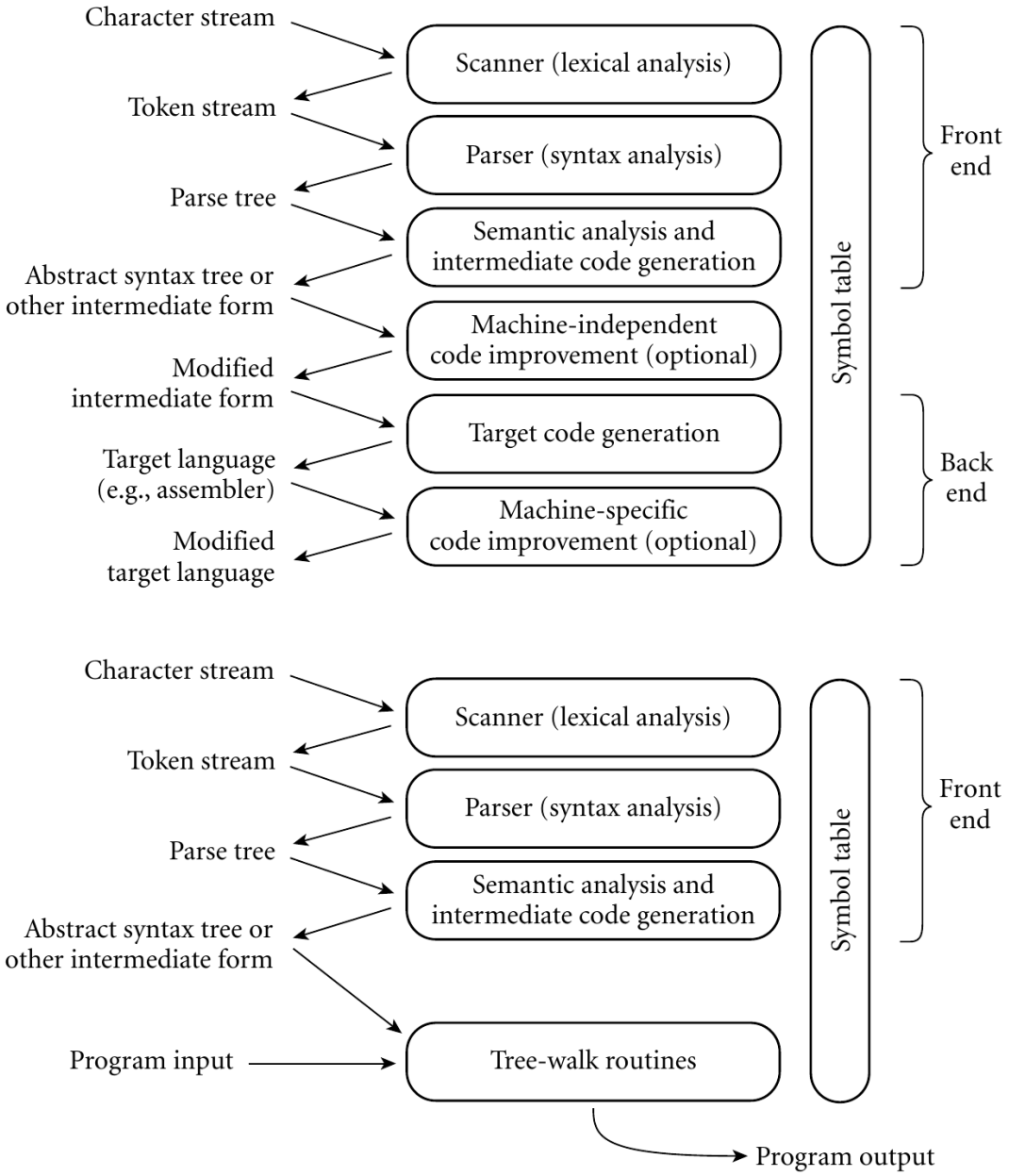
- el lexer creará e inicializará la tabla ya sea con los lexemas o sólo con un apuntador al identificador [1];
- el parser podrá distiguir las diferentes declaraciones de un mismo identificador, también puede redefinir la tabla para algún símbolo si es necesario;
- el analizador semántico podrá incluir más información sobre cada identificador para completar la tabla. Esta información podría ser el tipo de la variable, firmas, métodos y dependencias.

[*]Material revisado por el servicio social de Apoyo a la Docencia y Asesoría Académica en la Facultad de Ciencias de la UNAM con clave 2023-12/12-292, de Diana Laura Nicolás Pavia.

[1]Recordemos que un token es un par que contiene el nombre y un valor atributo. Este último es justo un apuntador a la tabla de símbolos de ese token.

El alcance de un identificador debe establecerse para delimitar el uso de un mismo identificador en diferentes partes del programa y posiblemente para diferentes usos. Por ejemplo, los contadores en algunos ciclos siempre son variables nombradas `i` o `j`, también es posible que algunos métodos de clase sean sustituidos por algunos otros de una superclase. Otro ejemplo son los bloques anidados en donde un mismo identificador se usa en varios bloques.

Tabla de Símbolos en compilación y en interpretación



Organización de una tabla de símbolos

La tabla de símbolos se puede ver como un diccionario: *mapea* nombres a información que el compilador ya conoce. Las operaciones principales que debe soportar la tabla son: inserción de nueva información,

buscar un identificador (*lookup*), además de manejar los alcances (*enter_scope* y *leave_scope*). Nunca se borra información de la tabla.

Anidar declaraciones en bloques puede manejarse al usar pilas que almacenen las tablas de símbolos creadas y así retener la declaración más cercana y establecer el alcance. Una forma común de implementar tablas de símbolos es usando las tablas hash, las cuales deben manejar funciones para respetar los alcances y restablecer información cada vez que se cierre o abra un alcance. En general, la organización de la tabla es vital para un mejor desempeño del compilador, veamos algunas formas de organización más usuales:

Lista Lineal

- *lookup* de orden lineal $O(n)$
- fácil de expandir ya que no tiene un tamaño fijo
- una asignación por inserción

Lista ordenada

- *lookup* de orden logarítmico $O(log_2\, n)$
- por cada inserción se debe reorganizar la lista

Árbol binario

- *lookup* lineal si el árbol no es balanceado y logarítmica si es balanceado
- fácil de expandir
- una asignación por inserción

Tabla Hash

- *lookup* constante en promedio
- expansión de la tabla depende del esquema usado

La tabla refuerza reglas o características del programa que no son capturadas por la estructura jerárquica de la gramática libre de contexto ni por el parse tree. Es así que durante el análisis semántico es posible agregar acciones semánticas a las producciones de la gramática para incluir información necesaria en la tabla de símbolos. Además, el análisis semántico también puede recuperar información de la tabla para decorar el árbol sintáctico. Alternativamente, un compilador también puede registrar las constantes usadas en el programa a través de una tabla de constantes.

Referencias

[1] Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. *Compilers, Principles, Techniques and Tools*. Pearson Education Inc., Second edition, 2007.

[2] Michael Lee Scott. *Programming Language Pragmatics*. Morgan-Kauffman Publishers, Third edition, 2009.

[3] Yunlin Su and Song Y. Yan. *Principles of Compilers, A New Approach to Compilers Including the Algebraic Method*. Springer-Verlag, Berlin Heidelberg, 2011.