La Tabla de Símbolos

La **tabla de símbolos** es una estructura de datos fundamental utilizada a lo largo de todo el proceso de compilación. Su función principal es almacenar y recuperar información sobre los identificadores del programa, como variables y funciones. Aunque en su forma más simple puede guardar pares de `<lexema, token>`, su rol más importante es manejar información semántica, como el tipo de dato de una variable.

El Desafío del Ámbito y su Solución SCOPE

Un problema central en la compilación es el manejo del **ámbito** (scope), que define dónde es visible y accesible un identificador. Los analizadores léxicos (*lexers*), basados en expresiones regulares, no son lo suficientemente potentes para gestionar ámbitos anidados (como bloques de código dentro de otros bloques).

Esta tarea más compleja recae sobre el analizador sintáctico (*parser*), que utiliza gramáticas libres de contexto, un formalismo más potente. Es el parser quien construye la verdadera tabla de símbolos que refleja la estructura de ámbitos del programa.

La solución para manejar ámbitos anidados es crear una **estructura de árbol de tablas de símbolos**. El funcionamiento es el siguiente:

- Al entrar en un nuevo ámbito (por ejemplo, un bloque `{}`), se crea una nueva tabla de símbolos.
- Esta nueva tabla apunta a la tabla del ámbito inmediatamente exterior que la contiene.
- Esta organización permite aplicar la regla del **símbolo más cercanamente anidado**: al buscar un identificador, se comienza en la tabla del ámbito actual y, si no se encuentra, la búsqueda continúa hacia las tablas exteriores hasta encontrar la declaración más cercana.

Interacción con la Tabla de Símbolos ⚙

La interfaz para manipular esta estructura de tablas de símbolos generalmente incluye tres operaciones clave:

1. **Crear tabla**: Genera una nueva tabla de símbolos para un nuevo ámbito, vinculándola a la tabla del ámbito exterior.

2. **Insertar entrada**: Añade un nuevo identificador y su información asociada (como su tipo) en la tabla del ámbito actual.

3. **Recuperar entrada**: Busca un identificador comenzando en la tabla actual y continuando hacia las tablas exteriores hasta encontrarlo, respetando así las reglas de ámbito.

Manejo de Palabras Clave Reservadas

Para manejar palabras clave del lenguaje (como `int` o `if`), estas se insertan en la tabla de símbolos **antes** de que el compilador comience a procesar el código fuente. De esta manera, cuando el lexer encuentra una de estas palabras, la tabla ya la reconoce. Esto permite identificar su uso correcto y marcar como error cualquier intento de utilizarlas como si fueran un identificador (por ejemplo, el nombre de una variable).

Aplicación Práctica a Través de Acciones Semánticas

La tabla de símbolos se manipula durante el análisis sintáctico mediante **acciones semánticas**, que son fragmentos de código incrustados directamente en las reglas de la gramática.

La **posición** de una acción semántica dentro de una regla de producción es crucial, ya que determina el momento exacto de su ejecución. El parser procesa el árbol de análisis en un recorrido de profundidad primero, por lo que las acciones se ejecutan en orden de izquierda a derecha.

- Una acción colocada **antes** de un símbolo en una regla se ejecutará antes de procesar dicho símbolo. Por ejemplo, una acción al inicio de la regla de un bloque se usaría para crear una nueva tabla de símbolos.

- Una acción al **final** de la regla de un bloque se ejecutaría al salir del mismo, permitiendo descartar la tabla de símbolos de ese ámbito.

El propósito final de esta estructura se manifiesta en reglas simples, como la que define un factor a partir de un identificador (`factor → id`). La acción semántica asociada a esta regla buscaría el `id` en la tabla de símbolos (respetando el ámbito actual) para recuperar su información, como su tipo, y así realizar la verificación semántica correcta.