UNIVERSIDAD INTERAMERICANA DE PANAMA

CONSTRUCCION TABLA DE SIMBOLOS

PROFESOR:

LEONARDO ESQUEDA

ALUMNA

CLARA INES BALANTA

FECHA

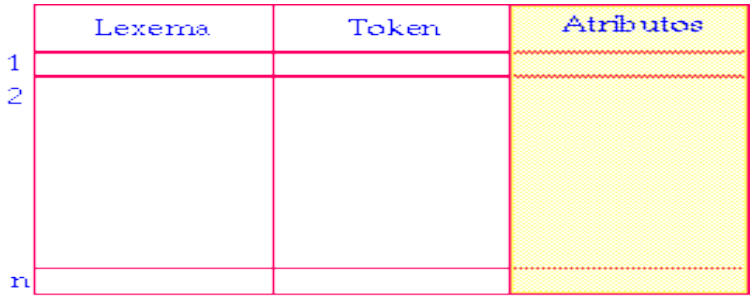
15 DE JULIO DE 2018

INTRODUCCION

El lenguaje es un vehículo por el cual se transmiten instrucciones a un procesador para que las ejecute y produzca ciertos resultados. Es tarea del compilador extraer el contenido semántico incluido en las sentencias del programa. Ciertos aspectos relativos a la corrección de un programa no se pueden expresar claramente mediante el lenguaje de programación. Es necesario dotar al compilador de rutinas auxiliares para captar todo lo que no se ha expresado mediante la sintaxis del lenguaje

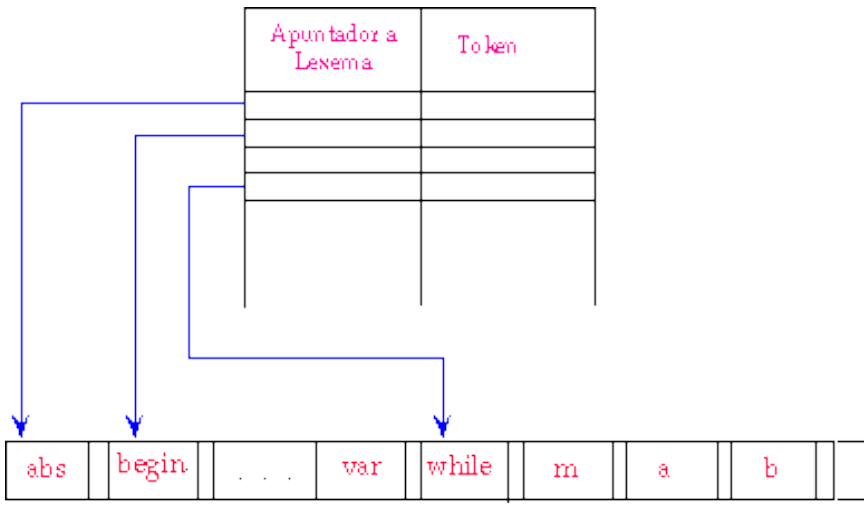
Las tablas de símbolos (también llamadas tablas de identificadores y tablas de nombres), realizan dos importantes funciones en el proceso de traducción: verificar que la semántica sea correcta y ayudar en la generación apropiada de código. Ambas funciones se realizan insertando o recuperando desde la tabla de símbolos los atributos de las variables usadas en el programa fuente. Estos atributos, tales como: el nombre, tipo, dirección de almacenamiento y dimensión de una variable, usualmente se encuentran explícitamente en las declaraciones o más implícitamente a través del contexto en que aparecen los nombres de variables en el programa.

Una de las estructuras de datos que se encuentran relacionadas con las fases del proceso de compilación es la tabla de símbolos, la cual tiene como propósito registrar información que se comparte entre varias etapas y que permite administrar los recursos asociados a las entidades que manipulará el programa. La tabla de símbolos tiene típicamente la siguiente estructura:



en donde podemos apreciar la designación de la entidad y su token -derivados del análisis de léxico- asi como una serie de atributos (tipo de dato, dirección en memoria) que emanan de otras fases (análisis gramatical y semántico). Las consultas a la tabla de símbolos se realizan por medio del lexema con que se designa a la entidad.

Esta concepción de la tabla de símbolos es demasiado simple para fines prácticos si consideramos que el lexema de la entidad es de longitud variable y se desea que la estructura sea homogenea. Una solución es considerar que en el campo lexema se tiene un apuntador (que siempre ocupa el mismo espacio) hacia donde se registrarán propiamente los lexemas. Eso evitará el desperdicio de memoria al tener el espacio justo para representar a cada lexema.



La creación de la tabla de símbolos compete inicialmente al analizador de léxico, quien registrará a las entidades (reconocidas bajo el patrón de Identificador) de manera única, por medio del binomio de operaciones Búsqueda-Inserción. En el contexto de un programa las entidades pueden describir propiamente objetos manipulables por el lenguaje (por ejemplo variables, constantes o funciones) o descriptores de acciones (las palabras reservadas); ambas situaciones son reconocidas bajo el mismo patrón de identificador y la tabla de símbolos se emplea para hacer su discriminación. El analizador de léxico funciona bajo el siguiente mecanismo:

REPITE Token = AnaLex(); SI Token == Identificador ENTONCES Token= REVISA_RESERVADAS(lexema) FSI HASTA QUE Token == EOF FREPITE

Como las palabras reservadas es un conjunto de entidades conocido y finito, la tabla de símbolos se inicializa con ellas y cuando se reconoce un identificador, su lexema se busca en la tabla y si se encuentra en ella, se regresa el token correspondiente a través de invocar a la función REVISA_RESERVADAS. El algoritmo de la funcion REVISA_RESERVADAS es el siguiente:

REVISA_RESERVADAS(lexema) INICIO p=BUSQUEDA(lexema) SI p == 0 ENTONCES p=INSERTAR(lexema) FSI Regresa TABLA[p].Token FIN

Este algoritmo supone al existencia de dos funciones que realizan la búsqueda y la inserción de un lexema en la tabla de símbolos. La última_entrada direcciona la última entidad registrada y último_lexema la primera localidad disponible donde se registran los lexemas.

La función BUSQUEDA se guía bajo el siguiente algoritmo:

BUSQUEDA (lexema) INICIO Indice=última_entrada MIENTRAS índice>0 REPITE Si contenido (TABLA[Indice].Ap_lexema)==lexema ENTONCES regresa Indice SINO Indice:=Indice-1 FSI FREPITE Regresa Indice FIN

donde es de notar que la búsqueda se inicia de la última entidad registrada hacia la primera bajo el supuesto que en un programa se encuentran con mayor frecuencia referencias a variables (que se registran al final) que a palabras reservadas (que estan registradas al inicio de la tabla).

Si la búsqueda fracasa, la entidad tiene que darse de alta por la función INSERTAR, cuyo algoritmo es:

INSERTAR (lexema) INICIO LEXEMAS[último_lexema] = lexema última_entrada = última_entrada + 1 TABLA[última_entrada].Ap_lexema = último_lexema. TABLA[última_entrada].Token = última entrada. último_lexema = último_lexema + LONGITUD(lexema). Regresa última_entrada FIN

Donde LONGITUD es una función que determina el número de caracteres que conforman al lexema insertado.

Cuando construir la tabla de símbolos y cuando interactuar con ella.

El punto del procesador de traducción en el cual son invocadas las rutinas de manejo de la tabla de símbolos dependen primeramente del número y la naturaleza de los pasos del compilador.

En un compilador multipasos, la tabla de símbolos es creada durante el paso de análisis léxico. Por medio de un índice se entra a la tabla de símbolos para ubicar la variable, a partir del token generado por el scanner.

Contenido de la tabla de símbolos.

Una tabla de símbolos puede conceptualizarse como una serie de renglones, cada uno de los cuales contiene una lista de valores de atributos que son asociados con una variable en particular. Las clases de los atributos que aparecen en una tabla de símbolos dependen en algún grado de la naturaleza del lenguaje de programación para el cual se escribe el compilador. Por ejemplo, un lenguaje puede ser sin tipos, y por lo tanto el atributo tipo no necesita aparecer en la tabla. Similarmente, la organización de la tabla de símbolos variará dependiendo de las limitaciones de memoria y tiempo de acceso.

A continuación, se presenta un ejemplo de una tabla de símbolos típica.

Número	Nombre Variable	Dirección	Tipo	Valor	Línea Declaración	Línea Referenciada	Liga
1	Empresa	0	2	1	2	9,14,25	3
2	X3	4	1	0	3	12,14	0
3	Formal	8	3	2	4	36,37,38	6
4	B	48	1	0	5	10,11,13,23	1
5	Resp	52	1	0	5	11,23,25	2
6	M	56	6	0	6	17,21	7
7	Primero	64	1	0	7	28,29,30,38	5

Los atributos que se manejan en la tabla anterior y que se describen enseguida, no son estrictamente necesarios para todos los compiladores; sin embargo, cada uno de tales atributos deberá ser considerado para la implementación de un compilador de un compilador en particular.

• Nombre de la variable.
• Dirección del código objeto.
• Tipo.
• Valor (o número de parámetros para uno procedimiento).
• Número de línea fuente donde fue declarada la variable.
• Números de línea fuente donde se hace referencia a la variable.
• Liga. Campo cuyos valores sirven para listar las variables en orden alfabético.

El nombre de la variable debe estar, en cualquier caso, formando parte de la tabla de símbolos, ya que es el medio por el cual una variable en particular es identificada

en las etapas de análisis semántico y generación de código. Para proveer un acceso rápido, es conveniente manejar un tamaño predefinido pero lo suficientemente grande para los nombres de las variables. Una longitud igual o mayor que 16 caracteres es bastante adecuado. El identificador completo puede almacenarse, justificado a la izquierda, en un campo de longitud fija en la tabla de símbolos. Este criterio posibilita un acceso más rápido a la tabla de símbolos, a costa de no aprovechar eficientemente el espacio de almacenamiento en el caso de las variables con identificadores cortos.

Otro criterio para manejar los identificadores en la tabla de símbolos consiste en colocar una cadena descriptora en el campo Nombre Variable de la tabla. El descriptor contiene los subcampos posición y longitud. El subcampo posición es un apuntador que indica la posición del primer carácter del nombre de la variable en un área general de cadenas, y el subcampo describe el número de caracteres del nombre de la variable. Este enfoque produce un acceso más lento a la tabla de símbolos, pero ofrece un ahorro considerable de espacio de almacenamiento.

En el proceso de compilación, una dirección de código objeto debe asociarse con todo variable en un programa. Esta dirección establece la ubicación relativa para los valores de la variable en tiempo de ejecución. La dirección del código objeto se coloca en la tabla de símbolos cuando la variable es declarada (o encontrada por primera vez). Esta dirección es reinvocada desde la tabla cuando la variable es referenciada en el programa fuente. La dirección es luego utilizada en una instrucción objeto que accesa (carga o almacena) el valor de esa variable.

El atributo tipo se almacena en la tabla de símbolos cuando los lenguajes compilados tienen tipos de datos explícitos o implícitos. Tradicionalmente, el tipo de dato de una variable es almacenado codificada en la tabla de símbolos; por ejemplo 1 puede representar el tipo real , 2 el tipo integer , etc.

Los atributos número_de_dimensiones y número_de_parametros son importantes en la fase de análisis semántico. En las referencias a los arreglos, el número de dimensiones debe coincidir con el especificado en la declaración del arreglo, y esto debe ser verificado en la fase de análisis semántico. El número de dimensiones se utiliza también como parámetro en el cálculo de la dirección de un elemento particular del arreglo. El número de parámetros de la invocación a un procedimiento también debe coincidir con el número usado en la declaración. En la construcción de la tabla de símbolos, es conveniente considerar el número de parámetros de un procedimiento como su número de dimensiones y así combinar estos dos atributos en uno. Además de conveniente, este enfoque también es consistente, ya que el tipo de verificación semántica para ambos atributos es similar.

El atributo liga se ha incluido en la tabla de ejemplo simplemente para facilitar la producción de un listado de referencias cruzadas ordenado alfabéticamente por nombre de variable.

Construcción tabla de símbolos

Una tabla de símbolos es una estructura de datos usada en el proceso de traducción de un lenguaje de programación (por un compilador o un intérprete), dónde cada símbolo en el código fuente de un programa está asociado con información tal como la ubicación, el tipo de datos y el ámbito de cada variable, constante o procedimiento.

Esta estructura de datos contiene una entrada o registro para cada identificador. Cada registro incluye los campos para los atributos del identificador. El administrador de la tabla de símbolos se encarga de manejar los accesos a la tabla de símbolos, en cada una de las etapas de compilación de un programa.

Se examina la tabla de símbolos cada vez que se encuentra un nombre en el texto fuente. Si dicho nombre ya existiese, se realizan cambios sobre la tabla existente. Un mecanismo de tabla de símbolos debe permitir añadir entradas nuevas y encontrar entradas existentes de manera eficaz.

La tabla de símbolos se construye durante el proceso de análisis. La información en la tabla de símbolos se utiliza tanto en el análisis (para especificar restricciones contextuales) como en la síntesis/traducción (para interpretar/traducir el significado de los tokens).

La información se reúne en las fases de análisis del compilador y la emplea la fase de síntesis para generar el código objeto. Por ejemplo, durante el análisis léxico, la cadena de caracteres, o lexema, que forma un identificador se guarda en una entrada de la tabla de símbolos. Las fases posteriores del compilador pueden añadir a esta entrada información, como el tipo del identificador, su uso (por ejemplo, procedimiento, variable o etiqueta) y su posición en la memoria. La fase de generación de código usaría después esta información para generar el código apropiado para almacenar y acceder a esta variable.

ESTRUCTURA
Cada entrada de la tabla de símbolos corresponde a al declaración de un nombre. El formato de las entradas no tiene que ser uniforme porque la información de un nombre depende del uso de dicho nombre. Cada entrada en principio puede considerarse:

· LEXEMA: distintas políticas de almacenamiento.
· ATRIBUTOS: cuanta información contiene dicho campo depende del objeto que denota el lexema.
La estructura inicial de la tabla de símbolos suele constar de dos partes:
· PARTE FIJA: formada por las palabras clave del lenguaje (suelen ser de uso reservado y del orden de unas decenas de palabras en un lenguaje programación típico)
· PARTE VARIABLE: definida por el programador: con el significado de los identificadores utilizados en cada frase (programa).
La información se introduce en la tabla de símbolos a la vez. Las palabras claves se introducen al inicio, o bien también podrían iniciarse en una tabla separada.

El analizador léxico busca secuencia de letras y dígitos en la tabla de símbolos para determinar si se ha encontrado una palabra clave reservada o un nombre, este es el motivo por el que las palabras reservadas deben estar en la tabla de símbolos antes de que comience el análisis.

Si se da el caso que el analizador léxico reconoce las palabras clave reservadas, entonces no necesitan aparecer en la tabla de símbolos puede tratarse como una estructura transitoria o volátil, que sea utilizada únicamente en el proceso de traducción de un lenguaje de programación, para luego ser descartada, o integrada en la salida del proceso de compilación para una explotación posterior, como puede ser por ejemplo, durante una sesión de depuración, o como recurso para obtener un informe de diagnóstico durante o después la ejecución de un programa.

La construcción de la tabla de símbolos debe seguirse de una correcta especificación de la misma. Como la definición de la parte dinámica de la tabla de símbolos está basada en las restricciones del lenguaje de programación, la especificación de la construcción de la tabla de símbolos está dirigida por la sintaxis de este sub-lenguaje de declaración y utiliza las mismas técnicas utilizadas para especificar y construir otros elementos del procesador de lenguaje (La construcción de la tabla de símbolos es una tarea del módulo de análisis sintáctico en la que coopera inicialmente el módulo de análisis léxico).

Asumimos que no hay área de declaraciones en nuestro lenguaje, por lo que la inserción de los símbolos en la tabla se debe hacer desde una acción léxica. Como atributo del token ID se usará un puntero a su entrada en la tabla de símbolos.
Las acciones semánticas usarán este puntero para acceder al valor de cada variable: si el identificador está a la izquierda del token de asignación, entonces se machacará el valor; y si forma parte de una expresión, ésta se evaluará al valor de la
variable.

La información que el desarrollador decida almacenar en esta tabla dependerá de las características concretas del traductor que esté desarrollando. Entre esta información puede incluirse:

• Nombre del elemento. El nombre o identificador puede almacenarse limitando o no la longitud del mismo.

• Tipo del elemento. Cuando se almacenan variables, resulta fundamental conocer el tipo de datos a que pertenece cada una de ellas, tanto si es primitivo como si no, con objeto de poder controlar que el uso que se hace de tales variables es coherente con el tipo con que fueron declaradas.

• Dirección de memoria en que se almacenará su valor en tiempo de ejecución. Esta dirección es necesaria, porque las instrucciones que referencian a una variable deben saber donde encontrar el valor de esa variable en tiempo de ejecución con objeto de poder generar código máquina, tanto si se trata de variables globales como de locales.

• Valor del elemento. Cuando se trabaja con intérpretes sencillos, y dado que en un intérprete se solapan los tiempos de compilación y ejecución, puede resultar más fácil gestionar las variables si almacenamos sus valores en la tabla de símbolos.

• Número de dimensiones. Si la variable a almacenar es un array, también pueden almacenarse sus dimensiones.

• Tipos de los parámetros formales. Si el identificador a almacenar pertenece a una función o procedimiento, es necesario almacenar los tipos de los parámetros formales para controlar que toda invocación a esta función sea hecha con parámetros reales coherentes.

CONCLUSIONES

Si se va a compilar programas con variables, se pueden usar tablas de símbolos no ordenados. Si el número de variables no supera a 25 se pueden usar tablas de símbolos ordenadas, utilizando el algoritmo de búsqueda binaria. Las tablas de símbolos con estructura de árbol se pueden utilizar en ciertos lenguajes con unas características muy determinadas (por ejemplo, BASIC y FORTRAN tienen limitados los nombres de los identificadores a unos pocos caracteres).

El mejor método cuando no se tienen problemas de memoria reducida es el direccionamiento hash abierto, su problema principal es la asignación de memoria para la tabla de símbolos. Esto en parte utilizando en el compilador algún parámetro que nos dé una idea a prioridad del tamaño de la tabla de símbolos (por ejemplo el número de líneas del programa). Otro método igualmente válido son los árboles y pilas.

BIBLIOGRAFIA

di002.edv.uniovi.es/~cueva/publicaciones/monografias/41_TablasDeSimbolos.pdf www.lcc.uma.es/~galvez/ftp/tci/tictema5.pdf

https://www.fdi.ucm.es/profesor/fpeinado/courses/.../Tema1.2-AtributosTS.pdf

www.cc.uah.es/ie/docencia/.../ProcesadoresDeLenguajeTema4_1xpagina.pdf https://es.slideshare.net/julietteleon71/compiladorestabla-de-simbolos