INSTITUTO TECNOLOGICO SUPERIOR DE CHICONTEPEC

Nombre:

Mayra Cruz Santiago.

Numero de control:

1717v0086

Materia:

Lenguajes y Autómatas 1

Docente:

Ing. Efrén Flores Cruz

Tema:

Análisis Léxico

Fecha de entrega:

11de Junio 2020

Unidad 5 Análisis Léxico

Lenguajes y autómatas 1

Unidad 5 Análisis Léxico

5.1 Funciones del Analizador, lexico

Es la primera fase de un compilador consistente en un programa que recibe como entrada el codigo fuente de otro programa (secuencia de caracteres) y produce una salida compuesta de tokens (componentes lexicos) o simbolos.

Token	Descripcion Informal	Lexemas de Ejemplo
If	Caracteres i, f	if
else	Caracteres, e, l, s, e	Else
Comparación	<0>0<=0>=0==0!=	<=, !=
id	letra seguida por letras y digitos	Pi, puntuacion, D2
numero	Cualquier Constante numérica	3.14159, 0, 6.02e23
literal	Cualquier cosa excepto ", rodeada por "s	"core dumped"

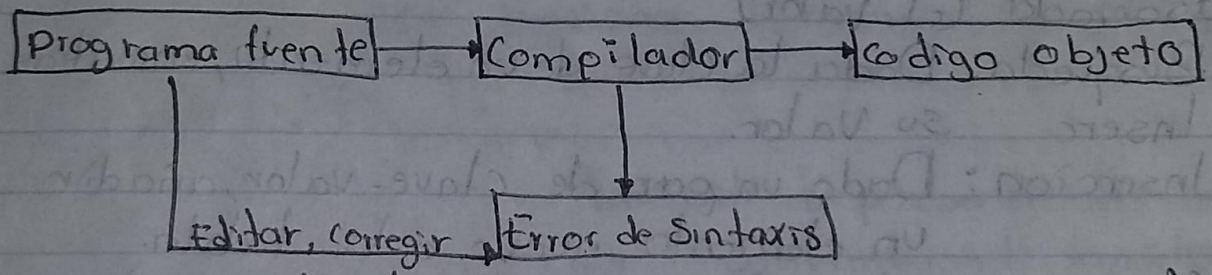
5.2 Componentes Lexicos, Patrones, Lexemas

Un token es un par que consiste en un nombre de token y un valor de atributo opcional. El nombre del token es un simbolo abstracto que representa un tipo de unidad lexica; por ejemplo, una palabra clave especifica o una secuencia de caracteres de entrada que denotan un identificador. Los nombres de los tokens son los simbolos de entrada que procesa el analizador sintactico. A partir de este momento, en general escribiremos el nombre de un token en negrita con frecuencia nos referiremos a un token por su nombre.

- Un patron. Es una descripción de la forma que pueden tomar los lexemas de un token, En el caso de una

palabra clave como token, el patrón es sólo la secuencia de caracteres que forman la palabra clave. Para los identificadores y algunos otros tokens, el patrón es una estructura más compleja que se relaciona mediante muchas cadenas.

- Un lexema: Es una secuencia de caracteres en el programa fuente, que coinciden con el patrón para un token y que el analizador léxico identifica como una instancia de ese token



Los componentes léxicos se suelen definir como un tipo enumerado. Se codifican como enteros. También se suelen almacenar la cadena de caracteres que se acaba de reconocer (el lexema) que se usará posteriormente para el análisis semántico.

```
typedef enum {TKN_IF, TKN_THEN, TKN_NUM, TKN_ID,...} TokenType;
```

Es importante conocer el lexema (para construir la tabla de símbolos). Los componentes léxicos se representan mediante una estructura registro con tipo de token y lexema.

```
typedef struct {
    TokenType token;
    Char *lexema; // Se reserva memoria dinamicamente
} TokenRecord;
TokenRecord getToken (void);
```

5.3 Creación de tabla de tokens

Tabla: Conjunto de pares clave-valor, llamados elementos de la tabla. La tabla de simbolos es un componente necesario de un compilador. Al declarar un identificador (normalmente una sola vez), éste es insertado en la tabla. Cada vez que se utilice el identificador se realizará una busqueda en la tabla. Cada vez que se utilice el identificador se realizará una búsqueda en la tabla para obtener la información asociada (el valor)

- Busqueda: Dada la clave de un elemento, encontrar su valor.
- Insercion: Dado un par de clave-valor, añadir un elemento nuevo a la tabla.
- Cambio de valor: Buscar el elemento y cambiar su valor.
- Borrado: Eliminar un elemento de la tabla.
- Longitud de búsqueda (o tiempo de acceso)

5.4 Errores Lexicos.

El analisis léxico constituye la primera fase, aqui se lee el programa fuente de izquierda a derecha y se agrupa en componentes léxicos (tokens), que son secuencia de caracteres que tienen un significado. Ademas, todos los espacios en blanco, lineas en blanco, comentarios y demas información innecesaria se elimina del programa fuente. Tambien se comprueba que los simbolos del lenguaje (palabras clave, operadores) se han escrito correctamente.

Como la tarea que realiza el analizador lexico es un caso especial de coincidencia de patrones, se necesitan los metodos de especificación y reconocimiento de patrones, y estos metodos son principalmente las expresiones regulares y los automatas finitos. Sin embargo, un analizador léxico tambien es la parte del tradoctor que maneja la entrada del Codigo fuente. puesto que esta entrada a menudo Involucra un Importante gasto de tiempo, el analizador léxico debe funcionar de manera tan eficiente como sea posible.

El compilador tiene que

1. Reportar clara y exactamente la presencia de errores.
2. Recuperarse de cada error lo suficientemente rapido para poder detectar errores subsiguientes.

- Tratar de evitar mensajes falsos de errores
- Un error que produce un token erróneo.
- Errores léxicos posibles

5.5 Generadores de Analizadores Léxicos

LEX : Codigo generado : C
FLEX : C++
ZLEX : C soporta códigos de caracteres de 16 bits
JAX : Java, no soporta entornos, basado en ER, no soporta Unicode
JLEX : Java, similar a lex, Diseñado para ser usado junto con CUP
JFLEX : Java, Diseñado para ser usado junto con Cup.

Se pueden usar varios técnicas para acelerar el algoritmo y ahorrar espacio. Las etiquetas pueden guardarse mediante dispersión para producir un sistema de firma que acelere sus busquedas. Como las tablas de transiciones son muy escasas, se pueden guardar en una lista corta que se consulte cada vez que se necesite hacer una transición a partir de un estado.
Estas listas no suelen tener mas de tres o cuatro elementos, asi que su busqueda sera razonablemente rapida.

5.6 Aplicaciones (caso de Estudio)

Además de, para construir compiladores e interpretes, los analizadores léxicos se pueden emplear para muchos programas convencionales.
Los ejemplos mas claros son aquellos programas que tienen algún tipo de entrada de texto donde hay un formato razonablemente en cuantos espacios y comentarios. En estos casos es bastante engorroso controlar donde empieza y termina cada componente y es facil liarse con los punteros a char. Un analizador léxico simplifica notablemente la interfaz y se dispone de un generador automático, el problema se resuelve en pocas lineas de codigo.