UNICEN
Universidad Nacional del Centro
de la Provincia de Buenos Aires

Diseño de Compiladores I
Resolución trabajo práctico 1 y 2

Grupo 6:

- Grecco, Gian:
  - LU: 248600
  - email: giangrecco.gg@gmail.com
- Lazarte, Marcos Alejandro:
  - LU: 248583
  - email: alelazarte55@gmail.com

Temas asignados

Práctico 1:

- **Enteros**: Constantes enteras con valores entre –2^(15) y 2^(15)–1. Estas constantes llevarán el sufijo "**_i**". Se debe incorporar a la lista de palabras reservadas la palabra **INTEGER**.
- **Flotantes**: Números reales con signo y parte exponencial. El exponente comienza con la letra f (minúscula) y llevará signo. La parte exponencial puede estar ausente. Ejemplos válidos:

  1. .6 -1.2 3.f–5 2.f+34 2.5f-1 15. 0.

  Considerar el rango 1.17549435f-38 < x < 3.40282347f+38 unión -3.40282347f+38 < x < -1.17549435f-38 unión 0.0
  Se debe incorporar a la lista de palabras reservadas la palabra **FLOAT**.

- Incorporar a la lista de palabras reservadas las palabras **WHILE** y **LOOP**.
- **Comentarios multilínea**: Comentarios que comienzan con "/%" y terminan con "%/" (estos comentarios pueden ocupar más de una línea).
- **Cadenas multilínea**: Cadenas de caracteres que comiencen y terminen con " "" . Estas cadenas pueden ocupar más de una línea, y en dicho caso, al final de cada línea, excepto la última,debe aparecer un guión " -". (En la Tabla de símbolos se guardará la cadena sin el guión,y sin el salto de línea).

Práctico 2:

- Tema 7 en TP1: Control de invocaciones recibidas - Pasaje de parámetros por copia valor resultado.

  Sentencias declarativas:

  - Modificar los encabezados de declaraciones de funciones con la siguiente estructura:

    PROC ID (<lista_de_parametros>) **NI=**n{ ... }

    Donde n será una constante de tipo entero (temas 1-2-3-4).

  - Modificar la estructura de cada parámetro, permitiendo usar la palabra reservada **VAR**, delante de cada parámetro.
  - Ejemplos de declaraciones de procedimiento válidas:

    PROC f1(**INTEGER** x, **VAR FLOAT** y, **FLOAT** z) **NI=** 2 {...}

    PROC f2(**VAR DOUBLE** a, **VAR DOUBLE** b) **NI=** 4 {...}

    // Incorporar **NI** y **VAR** a la lista de palabras reservadas.

    Sentencias ejecutables:

  - Sin cambios.
- Tema 15: **Sin conversiones**: Se explicará y resolverá en trabajos prácticos 3/4.(Grupos 2, 3, 6, 10, 11, 12, 15, 17,18, 19).

Introducción

En la resolución de los trabajos prácticos se realizó un analizador léxico y analizador sintáctico de un compilador. Se llevó a cabo por los temas que se describieron más arriba y por aclaraciones brindadas por la cátedra. La implementación es en java.

Para la realización del analizador léxico se hizo un diagrama de transición de estado, que después se representa en una matriz. Además se implementó una matriz de acciones semánticas.

En el analizador sintáctico se utilizó la herramienta Byacc para java que nos brindó un analizador sintáctico a partir de la gramática que desarrollamos. Se adaptó para que funciones con nuestro analizador léxico.

Desarrollo

Analizador Léxico

El analizador léxico es el encargado de reconocer los elementos del lenguaje usado (tokens), ajusta errores que son previstos avisando al programador con un “Warning”, también informa por algún “token” que se haya definido de forma incorrecta y almacena información de los “tokens” en una tabla de símbolos. Esta tabla es usada en la compilación.

Para llevar a cabo este analizador se tiene que considerar todos los caracteres que deben ser reconocidos para poder formar los “tokens”. Hay “tokens” que se consideran como palabras reservadas y estos no deben de ir en tabla de símbolo. Las constantes, identificadores y cadenas si deben de ir en la tabla de símbolo.

Diagrama Transición de Estados.

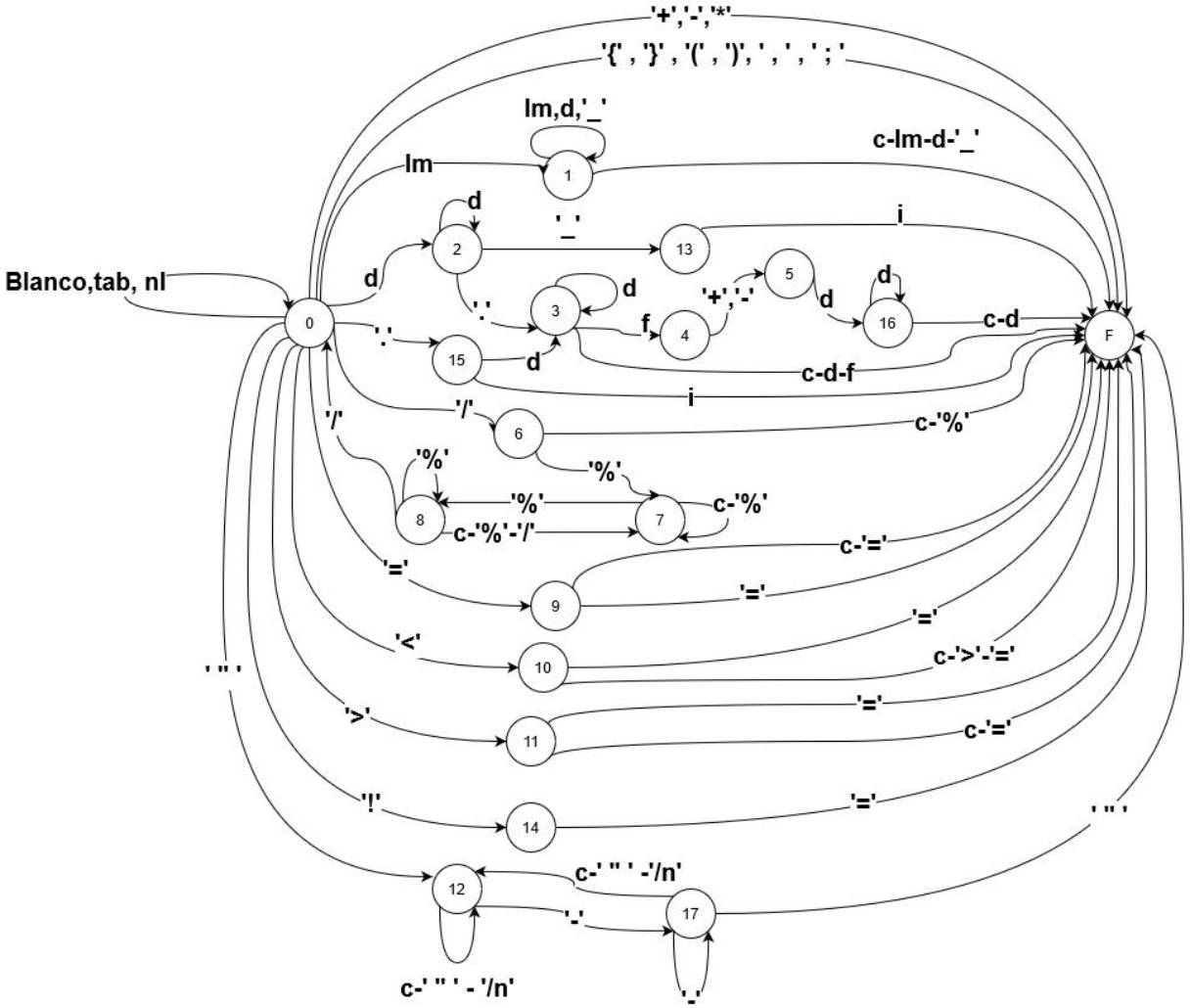

Diagrama de transición de estados

Aclaraciones sobre los símbolos que se ven en el diagrama:

- lm: letras
- d: dígito
- i: carácter ‘i’ para identificar las constantes Enteras de Tipo INTEGER

- f: carácter ‘f’ para interpretar los exponentes de Tipo FLOAT
- tab: tabulación
- nl: interpreta el salto de línea
- Blanco: interpreta un espacio en blanco

Matrices

Terminado el diagrama de transición de estado se realiza la matriz de transición de estado. Las filas indican el estado y las columnas un carácter que puede llegar. En la celda de tal fila y columna indica el siguiente estado que se ve en el autómata. Si no se describe una transición con su respectivo carácter, se debe a que hay un error.

La matriz de transición de estado permite visualizar caminos que no están contemplados en el diagrama. Para capturar cuando puede dar error.

	lm	**d**	**f**	**/**	*****	**+**	**-**	**=**	**<**	**>**	**{**	**}**	**(**	**)**	**,**	**;**	**"**	**.**	**%**	**_**	**Blanco tab**	**i**	**!**	**otro**	**nl**
0	1	2	1	6	F	F	F	9	10	11	F	F	F	F	F	F	12	15	E	E	0	1	14	E	0
1	1	1	1	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	1	F	1	F	F	F
2	E	2	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	3	E	13	E	E	E	E	E
3	F	3	4	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
4	E	E	E	E	E	5	5	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
5	E	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
6	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	7	F	F	F	F	F	F
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	8	7	7	7	7	7	7
8	7	7	7	0	7	7	7	7	7	7	7	7	7	7	7	7	7	7	8	7	7	7	7	7	7
9	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
10	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
11	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
12	12	12	12	12	12	12	17	12	12	12	12	12	12	12	12	12	F	12	12	12	12	12	12	12	E
13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	F	E	E	E
14	E	E	E	E	E	E	E	F	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
15	E	3	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	F	E	E	E
16	F	16	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
17	12	12	12	12	12	12	17	12	12	12	12	12	12	12	12	12	F	12	12	12	12	12	12	12	12

Matriz de transición de estados

A partir de lo anterior se crea la matriz de acciones semánticas. Estas acciones semánticas aplican acciones que ayudan al analizador léxico a controlar las pautas de desarrollo que aplicó la cátedra explicadas al principio del informe. En la matriz hay celdas que tienen “AS NU” es ya que en ese camino no es necesario una acción semántica, así que por eso se creó una acción semántica nula, no realiza ninguna ejecución.

	lm	d	f	/	*	+	-	=	<	>	{	}	(	)	,	;	"	.	%	_	Blanco tab	i	!	otro	nl
0	AS2	AS2	AS2	AS2	AS2	AS2	AS2	AS2	AS2	AS2	AS2	AS2	AS2	AS2	AS2	AS2	AS10	AS2	ASNU	ASNU	ASNU	AS2	AS2	ASNU	AS6
1	AS3	AS3	AS3	AS1	AS1	AS1	AS1	AS1	AS1	AS1	AS1	AS1	AS1	AS1	AS1	AS1	AS1	AS1	AS1	AS3	AS1	AS3	AS1	AS1	AS1
2	ASNU	AS3	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	AS3	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	AS6
3	AS5	AS3	AS3	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5
4	ASNU	ASNU	ASNU	ASNU	ASNU	AS3	AS3	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	AS6
5	ASNU	AS3	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	AS6
6	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS11	ASNU	AS7	AS7	AS7	AS7	AS7
7	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	AS6
8	ASNU	ASNU	ASNU	AS12	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	AS6
9	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS3	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7
10	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS3	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7
11	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS3	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7	AS7
12	AS3	AS3	AS3	AS3	AS3	AS3	AS3	AS3	AS3	AS3	AS3	AS3	AS3	AS3	AS3	AS3	AS9	AS3	AS3	AS3	AS3	AS3	AS3	AS3	AS6
13	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	AS4	ASNU	ASNU	AS6

14	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	AS3	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	AS6
15	ASNU	AS3	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	ASNU	AS6
16	AS5	AS3	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5	AS5
17	AS3	AS3	AS3	AS3	AS3	AS3	AS3	AS3	AS3	AS3	AS3	AS3	AS3	AS3	AS3	AS3	AS9	AS3	AS3	AS3	AS3	AS3	AS3	AS3	AS3	AS8

Matriz de acciones semánticas

Acciones semánticas

- Acción semántica 1
  * Chequea si es una palabra reservada si no,
  * chequea que la cantidad de caracteres sea menor a 20, si es mayor tira un Warning y trunca el ID.
  * Chequea que el ID sea en letras minúsculas, si no, tiramos warning y se transforma en minúscula luego.
  * Se fija en la tabla de símbolos por sí es un identificador ya es usado, si no existe se agrega .
  * Y vuelve un dígito para atrás.

- Acción semántica 2
  * Inicializa String (se reserva la máxima longitud permitida para identificadores).
  * Agrega letra a string.

- Acción semántica 3
  * Agrega al string el carácter, letra, dígito o guión bajo.
  * Agregar letra o dígito al string.

- Acción semántica 4
  * Chequea el límite superior de integer.
  * Se chequea en la tabla de símbolo.
  * Si no existe, se agrega.
  * La constante o mejor dicho lexema que guardamos como referencia en la tabla de símbolos no contiene '_i' ya que no se agregó en las acciones semánticas previas.

- Acción semántica 5
  * Se chequea los límites del float.
  * Se devuelve el último carácter a la entrada.
  * Se busca si se encuentra en la tabla de símbolos si no está, se agrega.
  * Donde lo que guardamos en la tabla de símbolos es el valor del float en forma de String donde el valor, es el valor real de la constante entera.

- Acción semántica 6
  * Agrega +1 al contador de línea.

- Acción semántica 7
  * Se devuelve el último carácter a la entrada.

- Acción semántica 8
  *Cuenta el salto de línea y elimina el "-" de la cadena.

- Acción semántica 9
  * Agregar el caracter al String.
  * Controla que se cierre la cadena.
  * Verifica que la cadena no se encuentre en la tabla de símbolos, si no esta la agrega.

- Acción semántica 10
  * Agregar el caracter al String.
  * Registra que se leyó una apertura de cadena.

- Acción semántica 11
  * Registra que se abre comentario.

- Acción semántica 12
  *Registra el cierre de comentario.

Implementaciones

Para las implementaciones hay distintos package, de Analizador Léxico, Semantico y las Acciones semánticas.

En el Package de Analizador léxico hay 4 clases, el mismo “AnalizadorLexico” donde se explica más adelante, la clase “Token” donde la mayor funcionalidad de esta es devolver el valor en código ASCII de las palabras reservadas o de algún identificador que se reconozca. Después se implementó la “TablaSimbolos” en una clase ya que luego se debe usar en el generador de código por ahora en la tabla de símbolos sólo guardamos el lexema que va a ser la key de la hash y un int que por ahora solo diferenciamos distintos tipos de constantes se puede hacer una clase contenedora en vez del int para poder almacenar datos necesarios en un futuro, y la clase “PalabrasReservadas” se definió como un contenedor con la posibilidad de agregar nuevas palabras en un futuro.

Luego en el Package Acciones Semántica se creó una clase abstracta con un único método execute() donde se define lo que hace cada una como parámetros necesitamos el analizador Léxico para poder utilizar funcionamiento de la misma, y el último carácter leído.

Analizador sintáctico

El analizador sintáctico tiene el deber de verificar que se cumplan las reglas sintácticas que se aclaran en la gramática. Se construyó una gramática con las peticiones del catedra a cada grupo. Con esta gramática y con la herramienta Byacc para Java se generó el analizador sintáctico.

Este analizador consume “tokens” que sirve para poder verificar si una secuencia en el programa pertenece al lenguaje o no.

En la gramática se definió la estructura válida para el lenguaje pero también las inválidas para informar al programador o advertir que lo ayudan a lograr un programa eficiente.

El compilador consume tokens hasta que termine de leer el archivo, este no para aunque haya errores, se lo advierte al programador y sigue compilando.

Se utilizaron los siguientes los no terminales:

- programa: inicio del programa.
- body: no terminal principal en donde se crean sentencias.
- sentencia: permite tener dos no terminales mas, ejecutable y declarativa.
- ejecutable: este admite la asignación, salida, control, selección, invocación.
- declarativa: controla la declaración de variables y procedimientos.
- procedure: controla la sintaxis de los procedimientos.
- dlistaparametros: verifica que la cantidad de parámetros no sea mayor a 3 variables.
- dparametro: controla los parámetros que se declaran en los procedimientos.
- listavariables: maneja las variables.
- asignacion: permite la construcción de una sentencia de asignación.
- salida: maneja la salida de la sentencia OUT.
- control: bucle WHILE.

- seleccion: sentencia IF.
- invocacion: admite la invocación de un procedimiento.
- condicion: construcción de condiciones para bucle WHILE y sentencia IF.
- comparador: define los comparadores permitidos.
- tipo: define los tipos que puede tener una variable.
- expresion: inicia las operaciones aritméticas.
- termino: permite construir operaciones aritméticas y mantiene procedencia.
- factor: para definir constantes, positivas y negativas.

Conclusión

Como primera parte al realizar el diagrama de transición de estados se puede apreciar las situaciones que hay que tener en cuenta, después con la matriz de transición de estados se visualiza caminos que en el diagrama no se tiene en cuenta y son los que se caracteriza como error. En la implementación del analizador léxico se profundizó más el funcionamiento de un compilador y las diferentes formas de como se puede implementar. Por esto se intentó mientras realizamos el trabajo implementarlo de una forma eficiente.

La herramienta Byacc facilita mucho en el desarrollo de un analizador sintáctico. Después de tener el analizador sintáctico hay que lograr que trabaje con el analizador léxico y así se puede compilar un programa. La gramática también facilita mucho ver errores de sintaxis.

El trabajo se desarrolló en el lenguaje Java, y este lenguaje es simple para este tipo de trabajos ya que no hay que controlar los punteros como si se programara en C++.