Universidad Tecnológica Nacional
Facultad Regional Buenos Aires

Sintaxis y Semántica De Los Lenguajes 2024

Curso: k2005
Turno: Viernes Mañana

TRABAJO PRÁCTICO ANUAL

GRUPO N° 1	
Nombre y Apellido	**Nombre y Apellido**
Santiago Carlos Boyadjian	Juan Cruz Miliani De Marval
Nicolas Agustin Caceres	Felipe Pampin
Franco Agustin Duarte	NicolÁs Francisco Reinoso
Manuel Liniers	Emanuel Santoro
Ludmila Belen Lolini	Luca Gonzalo Trias Lupincacci

FECHA DE PRESENTACIÓN 1:	Agosto 16 2024
FECHA DE PRESENTACIÓN 2:	Septiembre 13, 2024
FECHA DE PRESENTACIÓN 3:	Octubre 18, 2024
FECHA DE PRESENTACIÓN 4:	Noviembre 08, 2024

Sintaxis y Semántica del Lenguaje 2024

Trabajo Práctico II

Objetivo: Aplicar los conocimientos teóricos en el diseño de un compilador básico de Lenguaje de Programación.

Para ello, cada equipo, definirá un lenguaje básico, compuesto mínimamente:

- Uno o dos tipo de dato
- Identificadores con longitud máxima de 16 caracteres.
- Los identificadores deberán tener una convención predefinida
- Mínimo 2 sentencias, debiendo contar con convención de fin de sentencia , y de apertura y cierre.
- Cuatro (4) palabras reservadas
- Alfabeto a elección del equipo.

Primer Entrega: Gramática Léxica

a) Diseñar las categorías léxicas correspondiente al lenguaje elegido

b) Formalizar cada lenguaje regular con Autómatas Finitos y Expresiones regulares

1- Tipo de datos
Cadenas
Enteros

2- Identificadores
Las cadenas deben comenzar con "_" y deben ser continuadas con caracteres del alfabeto [a..z] o [A..Z] . Mínimo un carácter, y máximo 15
Los enteros deben comenzar con 'E' y deben ser seguidos de valores pertenecientes al alfabeto [0..9] . Mínimo un dígito, máximo 16. Ej E012345678901234, E0, E1294

ER cadenas = $\{(\_ . ([a..z] + [A..Z])^n \quad / \quad 0<n<16) \quad n \in Z\}$
ER enteros = $\{(E . [0..9]^n / 0<n<16) n \in Z\}$

3- Sentencias
Asignación: " -> " variable -> valor /
Impresión: "show" show(variable) /
Convención de fin de sentencia: " / "
Convención de apertura y cierre: [ ]

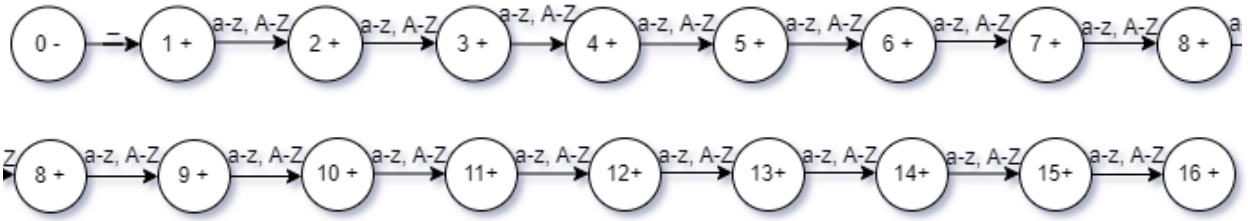
4- Palabras reservadas
cad
ent

show
si
ER = cad+ent+s(how+i)

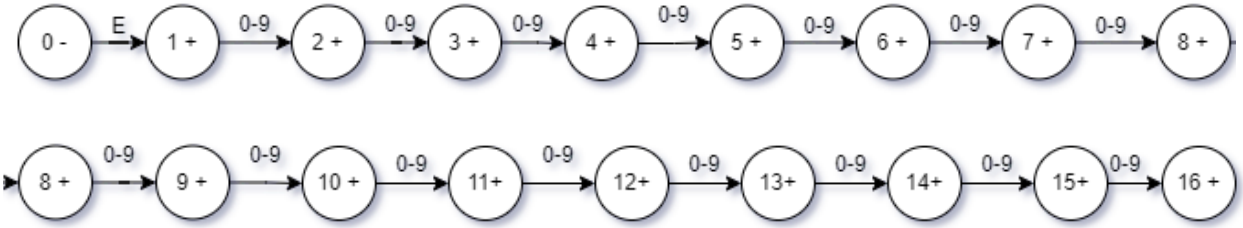
5- Alfabeto
Letras: [a..z] [A..Z]
Numeros: [0..9]
Símbolos: _, ->, / , [,].

Autómatas finitos:

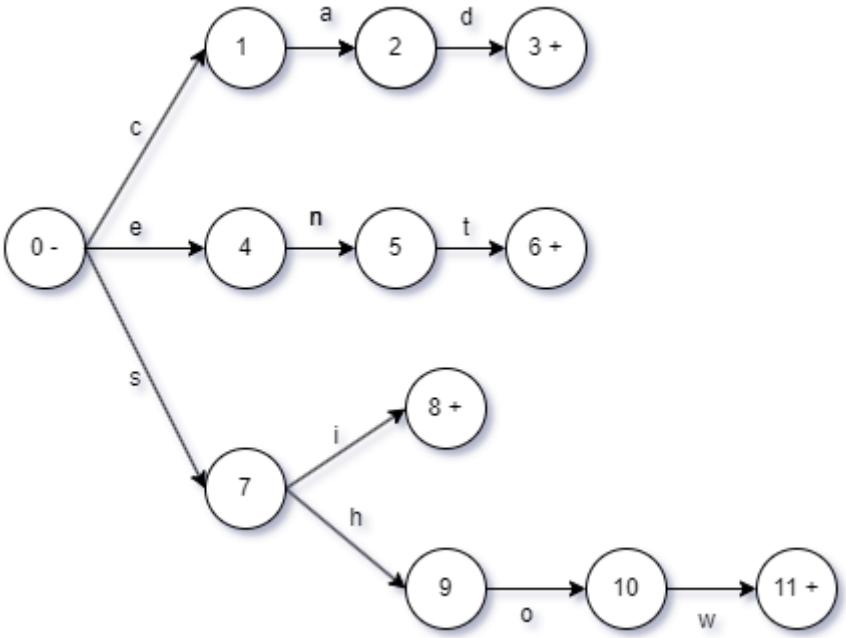
- Cadenas:



- Enteros:



- Palabras reservadas:



Segunda Entrega: Gramática de Estructura de Frases

a) Diseñar categorías sintácticas que se relacionen entre si y que hagan uso de las categorías léxicas de la Primer Entrega.

b) Formalizar cada lenguaje independiente de Contexto con BNF

Categorías sintácticas:

Expresiones: cadenas enteros identificadores

Instrucciones: palabraReservada

Sentencias: asignación impresión convencionFinSentencia convencionAperturaCierre

Formalización de cada lenguaje independiente de Contexto con BNF:

<sentencia> ::= <asignacion> | <impresion>

<asignacion> ::= <variable> "->" <valor> "/"

<impresion> ::= "show(" <variable> ")" "/"

<variable> ::= <cadena_id> | <entero_id>

<valor> ::= <cadena> | <entero>

<cadena_id> ::= "_" <letra> {<letra>}*

<entero_id> ::= "E" <digito> {<digito>}*

<cadena> ::= "\"" {<letra>}* "\""

<entero> ::= <digito> {<digito>}*

<letra> ::= [a..z] | [A..Z]

<digito> ::= [0..9]

<expresiones> ::= <identificador>

<identificador> ::= <digito> | <letra>

<digito> ::= [0..9]

<letra> ::= [a..z][A..Z]

<instrucciones> ::= <palabraReservada> | <caracterPuntuacion> | <identificador>

<palabraReservada> ::= <cad> | <ent> | <show> | <si>

<caracterPuntuacion> ::= “_” | “>” | “/” | “[ ]”

<identificador> ::= <digito> | <letra>

<digito> ::= [0..9]

<letra> ::= [a..z][A..Z]

Tercer Entrega: Scanner

- Desarrollar un scanner en Flex para las categorías léxicas
- Implementar argumentos de la línea de comando (programa comando)

Repositorio github: https://github.com/EmaSantoro/UTN-SSL-GRUPO1-TP

Código programa Scanner en Flex

```
/*
 * Sample Scanner1:
 * Description: Replace the string "username" from standard input
 *              with the user's login name (e.g. lgao)
 * Usage: (1) $ flex sample1.lex
 *        (2) $ gcc lex.yy.c -lfl
 *        (3) $ ./a.out
 *            stdin> username
 *        stdin> Ctrl-D
 */


%%
cad|s(how|i)|ent {printf("PALABRA RESERVADA: %s\n", yytext);}

_[a-zA-Z]{1,15} {printf("\nIDENTIFICADOR: %s\n", yytext);}

E[0-9]{1,16} {printf("IDENTIFICADOR: %s\n", yytext);}

-> {printf("OPERADOR: %s\n", yytext);}

"\["|"\]"|"\/" {printf("SIGNO PUNTUACION: %s\n", yytext);}

[0-9]{2,16} {printf("ENTERO: %s\n", yytext);}

([a-z]|[A-Z]){2,16} {printf("CADENA: %s\n", yytext);}

[a-zA-Z]|[0-9] {printf("CARACTER: %s\n", yytext);}


%%
int main(int argc, char **argv)
{
    yylex();
}

int yywrap(){}
```

Archivo Test

```
a
b
c
9

E1 -> 12/
_string -> [hola]

cadena


show
si
cad
ent
```

Validación en consola

```
lucatrias@lucatrias-MS-7C09:~/SySlTp$ ./a.out < ./test.txt
CARACTER: a

CARACTER: b

CARACTER: c

CARACTER: 9


IDENTIFICADOR: E1
 OPERADOR: ->
 ENTERO: 12
SIGNO PUNTUACION: /


IDENTIFICADOR: _string
 OPERADOR: ->
 SIGNO PUNTUACION: [
CADENA: hola
SIGNO PUNTUACION: ]


CADENA: cadena



PALABRA RESERVADA: show

PALABRA RESERVADA: si

PALABRA RESERVADA: cad

PALABRA RESERVADA: ent
```