
# UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

## FACULTAD DE INGENIERÍA

## Compiladores

**Espinoza González Isaac**
**Montoya Pérez Hector**
**Soto Vázquez Patricia**
**Pérez Dublán Juan Pablo**

## Programa 1(analizador léxico)

Universidad Nacional Autónoma de México
Semestre 2020-2
Compiladores
Analisis Lexico
Profesor: Adrián Ulises Mercado Martíinez
Programa 1

1. Para la gramática siguiente elaborar el analizador léxico utilizando flex.
2. Entregar un documento con lo siguiente
   (a) Análisis del problema
      i. Descripción del problema no del programa

Realizar un analizador léxico que va a validar un programa en pseudocodigo.

   (b) Diseño de la solución
      i. Separar los terminales de los no terminales

| | Terminales | No terminales |
|---|---|---|
| 1 | estructura | Programa |
| 2 | inicio | Declaraciones |
| 3 | fIn | Tipo _Registro |
| 4 | ent | Tipo |
| 5 | real | Base |
| 6 | dreal | Tipo_arreglo |
| 7 | car | Lista_var |
| 8 | sin | Funciones |
| 9 | ( | Argumentos |
| 10 | num | Lista_arg |
| 11 | ) | arg |
| 12 | id | Tipo_arg |
| 13 | def | param_arr |
| 14 | si | sentencias |
| 15 | entonces | sentencia |
| 16 | mientras | Casos |
| 17 | sino | Predeterminado |

| | | |
|---|---|---|
| 18 | hacer | e_bool |
| 19 | segun | relacional |
| 20 | escribir | oprel |
| 21 | devolver | expresion |
| 22 | leer | oparit |
| 23 | terminar | variable |
| 24 | caso | dato_est_sim |
| 25 | pred | arreglo |
| 26 | : | parametros |
| 27 | o | lista_param |
| 28 | y | |
| 29 | no | |
| 30 | verdadero | |
| 31 | falso | |
| 32 | cadena | |
| 33 | % | |
| 34 | + | |
| 35 | - | |
| 36 | * | |
| 37 | / | |
| 38 | ε | |
| 39 | > | |
| 40 | < | |
| 41 | >= | |
| 42 | <= | |
| 43 | <> | |
| 44 | == | |
| 45 | ; | |
| 46 | [ | |
| 47 | ] | |
| 48 | caracter | |

| | | |
|---|---|---|
| 40 | , | |

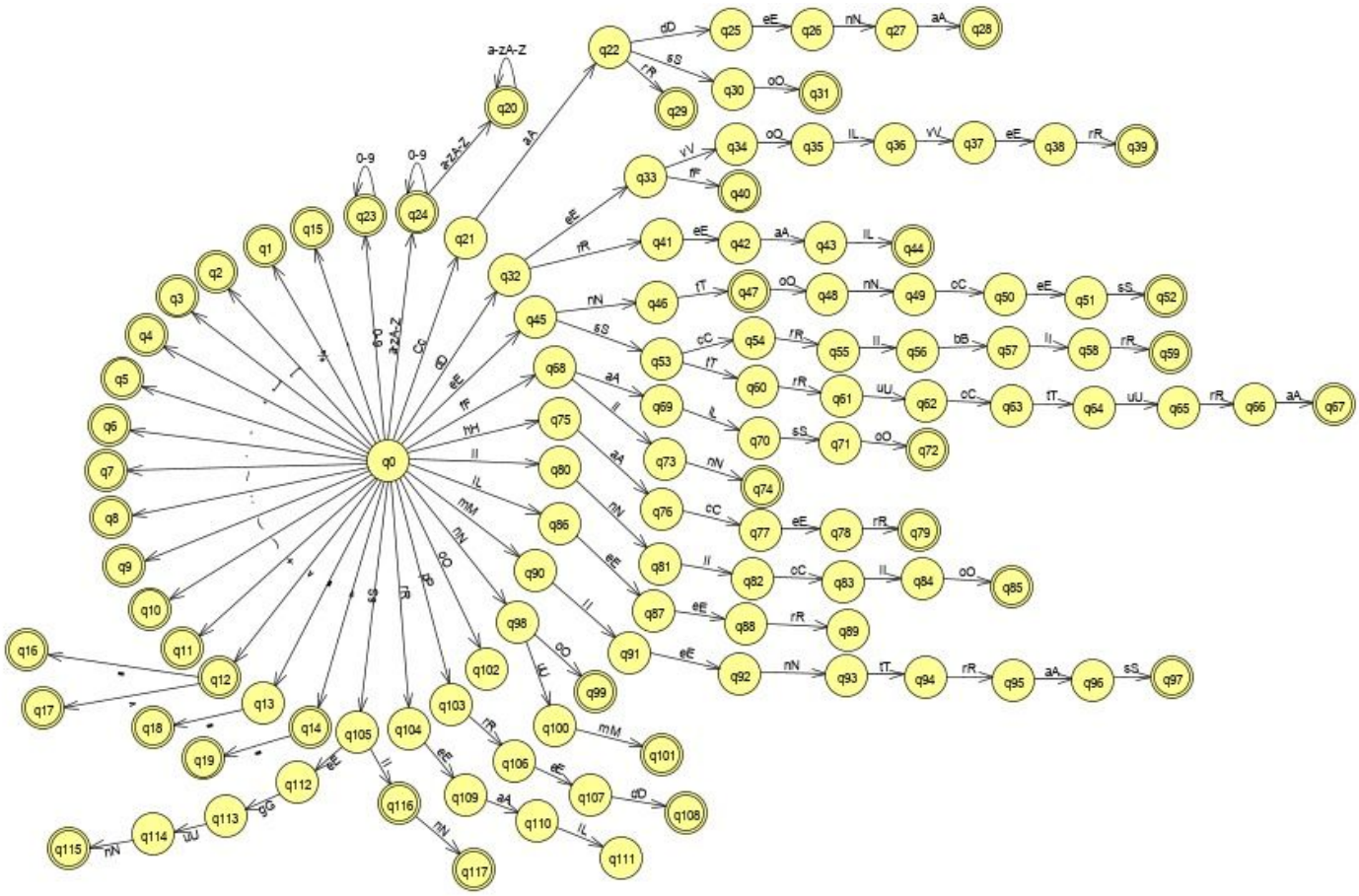
ii. Las expresiones regulares para los terminales

| |
|---|
| estructura→ [eE][sS][tT][rR][uU][cC][tT][uU][rR][aA] |
| inicio→ [iI][nN][iI][cC][iI][oO] |
| fIn→ [Ff][Ii][Nn] |
| ent→ [Ee][Nn][Tt] |
| real→ [Rr][Ee][Aa][Ll] |
| dreal→ [Dd][Rr][Ee][Aa][Ll] |
| car→ [Cc][Aa][Rr] |
| sin→ [Ss][In][Nn] |
| (→ [(] |
| num→ [0-9]+ |
| )→ [)] |
| id→ [a-zA-Z_][a-zA-Z_0-9] |
| def→ [Dd][Ee][Ff] |
| si→ [Ss][Ii] |
| entonces→[Ee][Nn][Tt][Oo][Nn][Cc][Ee][Ss] |
| mientras→ [Mm][Ii][Ee][Nn][Tt][Rr][Aa][Ss] |
| sino→ [sS][iI][nN][oO] |
| hacer→ [hH][aA][cC][eE][rR] |
| segun→ [sS][eE][gG][uU][nN] |
| escribir→ [eE][sS][cC][rR][iI][bB][iI][rR] |
| devolver→ [dD][eE][vV][oO][lL][vV][eE][rR] |
| leer→ [lL][eE][eE][rR] |
| terminar→ [Tt][Ee][Rr][Mm][Nn][Aa][Ll] |
| caso→ [Cc][Aa][Ss][Oo] |
| pred→ [Pp][Rr][Ee][dD] |
| :→ : |
| caracter→ ‘[a-zA-Z]’ |
| o→ [oO] |

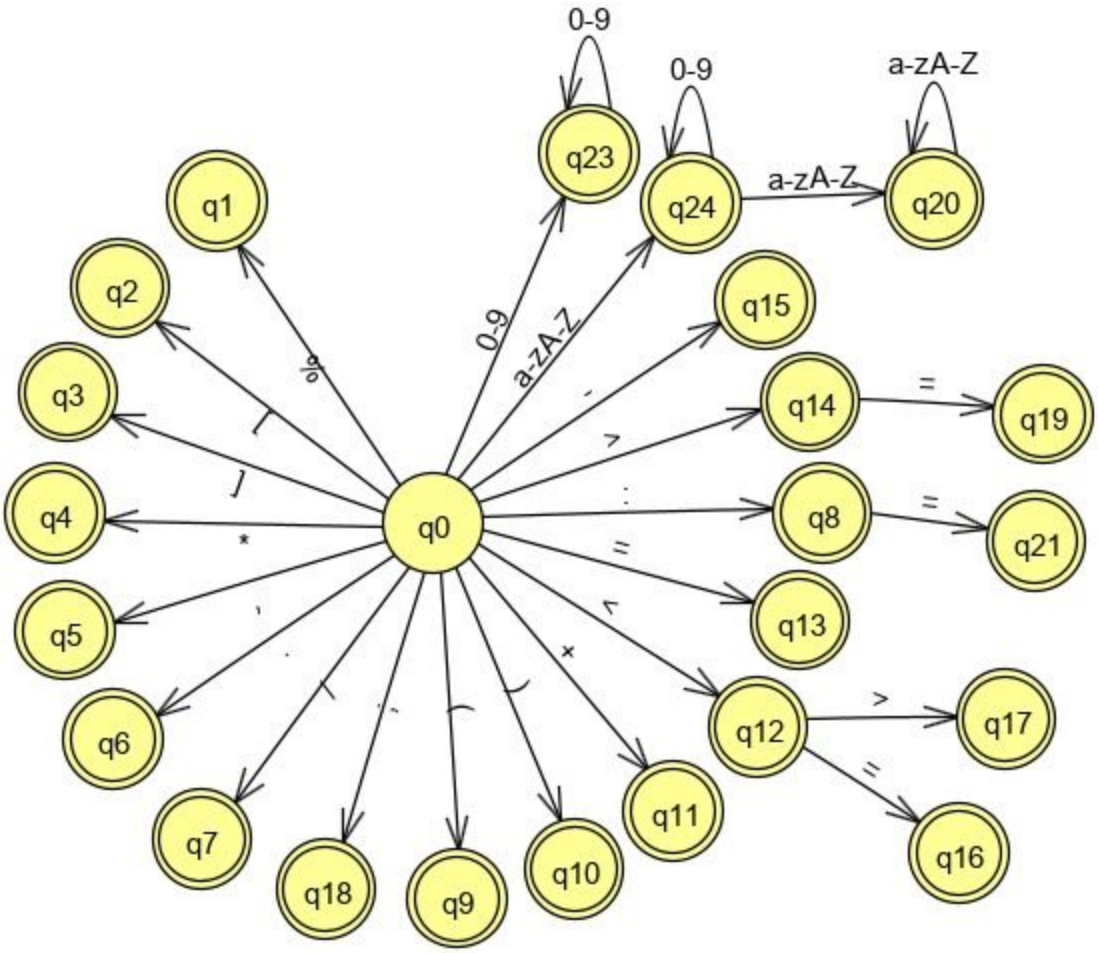
| |
|---|
| cadena→ “[a-zA-Z][a-zA-Z]+” |
| y→ [Yy] |
| no→ [Nn][Oo] |
| verdadero→ [Vv][Ee][Rr][Dd][Aa][Dd][Ee][Rr][Oo] |
| falso→ [Ff][Aa][Ll][Ss][Oo] |
| cadena→ [Cc][Aa][Dd][Ee][Nn][Aa] |
| %→ |
| +→”+” |
| -→ |
| *→”*” |
| /→/ |
| >→> |
| <→< |
| >=→>= |
| <=→<= |
| <>→<> |
| ==→== |
| ;→; |
| ,→, |
| [→[ |
| ]→] |
| .→ . |

iii. El AFD resultante(Imagen)

Del AFND



Obtuvimos el AFD




(c) Implementación. Describir cómo está implementado su programa, las partes que lo componen (no es todo el código).

Dentro de la simbología: %{ /* Código en C/* }% se encuentran las declaraciones realizadas en lenguaje C.

En el encabezado del programa se encuentra declaradas las bibliotecas:

```
{%
#include <stdio.h>
#include <stdlib.h>
```

```
#include <string.h>
%}
```

Posteriormente se definen las expresiones regulares para palabras reservadas, operadores, signos de puntuación y símbolos.

En la tercera parte del programa se incluyen los procedimientos auxiliares que nos permiten imprimir en pantalla cuando se identifique algún símbolo o palabra reservada según las expresiones regulares definidas anteriormente.

Finalmente se incluye la función error que indica la existencia de algún error léxico.

(d) Forma de ejecutar el programa.
./lexico.exe archivo

(e) Se debe incluir un archivo main.c donde se encuentre la función principal para ejecutar el programa.

(f) Cada función tiene que estar documentada con la fecha y persona que la programó. Además de una muy breve descripción de lo que hace la función.

(g) En caso de hacer cambios en alguna función agregar fecha de modificación y nombre de quien la modificó.

**Gramática**

sin: significa sin tipo, car: tipo carácter

1. programa → declaraciones funciones
2. declaraciones → tipo lista var; declaraciones
   | tipo registro lista_var; declaraciones
   | *ε*
3. tipo_registro → estructura inicio declaraciones fin
4. tipo → base tipo arreglo
5. base → ent | real | dreal | car | sin
6. tipo_arreglo → [num] tipo arreglo | *ε*
7. lista var → lista_var, id | id
8. funciones → def tipo id(argumentos) inicio declaraciones sentencias fin funciones | *ε*
9. argumentos → listar arg | sin
10. lista arg → lista arg, arg | arg
11. arg → tipo arg id
12. tipo_arg → base param arr

13. param arr → [ ] param arr | $\varepsilon$



14. sentencias → sentencias sentencia | sentencia

15. sentencia → si e bool entonces sentencia fin
    | si e bool entonces sentencia sino
    sentencia fin | mientras e bool hacer
    sentencia fin
    | hacer sentencia mientras e bool;
    | segun (variable) hacer casos predeterminado fin
    | variable := expresion ;
    | escribir expresion ;
    | leer variable ; | devolver;
    | devolver expresion;
    | terminar;
    | inicio sentencias fin

16. casos → caso num: sentencia casos | caso num: sentencia

17. predeterminado → pred: sentencia | $\varepsilon$

18. e bool → e bool o e bool | e bool y e bool | no e bool | ( e bool )
    | relacional | verdadero | falso

19. relacional → relacional oprel relacional | expresion

20. oprel → > | < | **>=** | **<=** | <> | **=**

21. expresion → expresion oparit expresion
    | expresion % expresion | (expresion ) | id
    | variable | num | cadena | caracter | id(parametros)

22. oparit → **+** | **−** | $*$ | /

23. variable → dato est sim | arreglo

24. dato est sim → dato est sim .id | id

25. arreglo → id [ expresion ] | arreglo [ expresion ]

26. parametros → lista param | $\varepsilon$

27. lista param → lista param, expresion | expresion