INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Título:

Práctica 4. Analizador Léxico.

Alumno:

Rojas Zepeda Luis Eduardo

Boleta:

2015090668

Asignatura:

Compiladores

Profesor:

Edgardo Adrián Franco Martínez

Maestro en ciencias de la computación.

Grupo:

3CM3

12/11/20

Introducción

En este proyecto de la asignatura de compiladores, con el fin de poner en práctica los conocimientos adquiridos durante el semestre, se realizará un compilador básico para el lenguaje C, el cual se apoyará de la especificación de las clases léxicas y expresiones regulares mostradas en el libro “El lenguaje de programación C” de Brian W. Kernighan y Dennis M. Ritchie, y de la especificación de Jutta Degener para Lex que se encuentra pública en internet. Además, se usará la herramienta flex, que según su propia definición es un “generador de analizador léxico rápido”, para la primera parte del compilador, en su versión 2.5.35 para el sistema operativo macOS.

La intención del proyecto es alcanzar un mayor nivel de comprensión de las unidades temáticas de la materia y aclarar dudas que puedan surgir en la aplicación de la teoría. Primeramente, se había pensado como proyecto un generador y sintetizador de señales digitales para generar tonos como los de los sintetizadores de los años 80’s llamados 808, utilizando un lenguaje propio creado, sin embargo, por diversas cuestiones se decidió iniciar con un proyecto más general como un compilador de C, para obtener experiencia y habilidades, y posiblemente en un futuro llevar acabo un proyecto con mayor nivel de dificultad como el del sintetizador.

Desarrollo

Metodología:

Ejemplificación del lenguaje

```
113  void burbujaOpti(int *A, int n){
114
115    int i, j; //Variables para loops
116    int aux; //Variable de apoyo para swapping
117
118    for(i=0;i<=(n-2);i++){
119      for(j=0;j<=((n-2)-i);j++){
120        if(A[j]>A[j+1]){
121          aux=A[j];
122          A[j]=A[j+1];
123          A[j+1]=aux;
124        }
125      }
126    }
127
128    for(i=0;i<n;i++){
129      printf("%i \n",A[i]);
130    }
131  }
```

Programa de ordenamiento.

Clases léxicas

Descripción	Símbolo	Definición
Dígito	D	`[0-9]`
Letras	L	`[a-zA-Z_]`
Hexadecimales	H	`[a-fA-F0-9]`
Notación científica	E	`[Ee][+-]?{D}+`
Sufijo	FS	`(f|F|l|L)`
Sufijo	IS	`(u|U|l|L)*`

Expresiones regulares

Descripción	Símbolo
Inicio de comentario	/*
Palabras reservadas	"auto" "break" "case" "char" "const" "continue" "default" "do" "double" "else" "enum" "extern" "float" "for" "goto" "if" "int" "long" "register" "return" "short" "signed" "sizeof" "static" "struct" “switch" "typedef" "union" "unsigned" "void" "volatile" “while"
Tipos de datos	{L}({L}|{D})*
Constantes	0[xX]{H}+{IS}? 0{D}+{IS}? {D}+{IS}? L?'(\\.|[^\\'])+' {D}+{E}{FS}? {D}*"."{D}+({E})?{FS}? {D}+"."{D}*({E})?{FS}?
Cadenas	L?\”(\\.|[^\\"])*\"

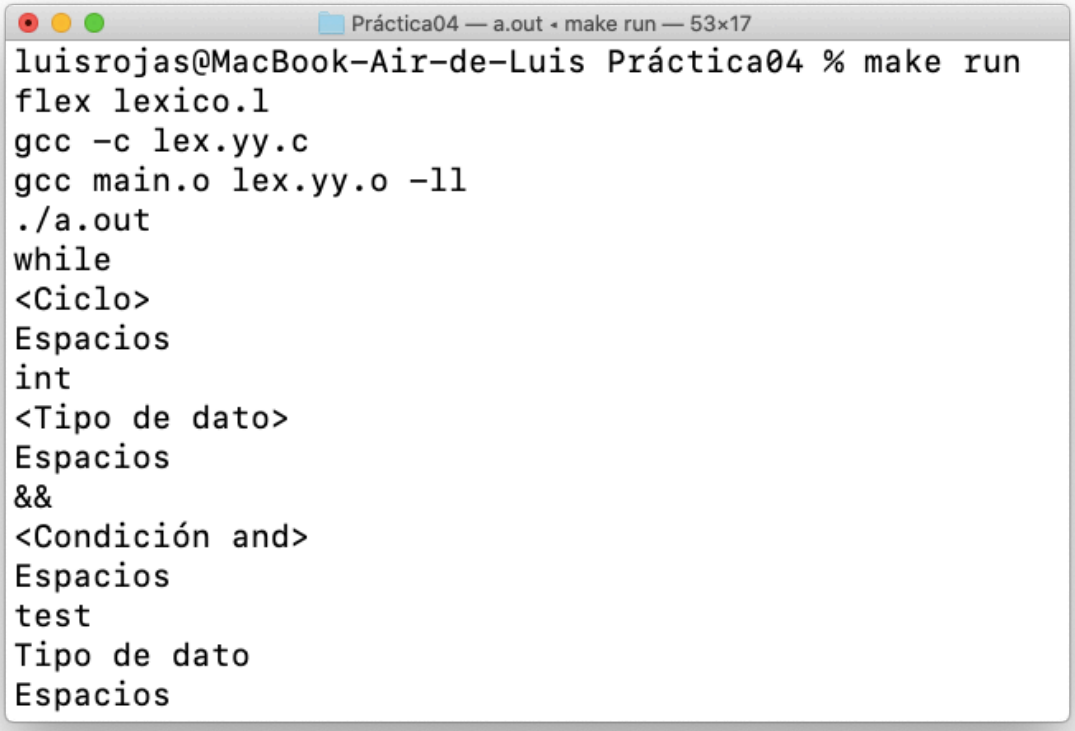
Operadores	"..." ">>=" "<<=" "+=" "-=" "*=" "/=" "%=" "&=" "^=" "|=" ">>" "<<" "++" "--" "->" "&&" "||" "<=" ">=" "==" "!=" ";" ("{"|"<%") ("}"|"%>") "'" ":" "=" "(" ")" ("["|"<:") ("]"|":>") "." "&" "!" "~" "-" "+" "*" "/" "%" "<" ">" "^" "|" "?"
Espacios	[\t\v\n\f]
Caracteres ignorados	.

Codificación

```
%{
  /* Definiciones */
  /*Código C*/
  #include <stdio.h>
%}
D     [0-9]
L     [a-zA-Z_]
H     [a-fA-F0-9]
E     [Ee][+-]?{D}+
FS    (f|F|l|L)
IS    (u|U|l|L)*
%%
"/*"          { printf("<Salto de linea>\n"); }

"auto"        { printf("<Especificador de categoria>\n"); }
"break"       { printf("<Terminar ciclo>\n"); }
"case"        { printf("<Especificador de caso>\n"); }
"char"        { printf("<Tipo de dato>\n"); }
"const"       { printf("<Calificador de constante>\n"); }
```

```
"volatile"  { printf("<Palabra reservada>\n"); }
"while"     { printf("<Ciclo>\n"); }

{L}({L}|{D})*   { printf("Tipo de dato\n"); }

0[xX]{H}+{IS}?    { printf("Constante hexadecimal\n"); }
0{D}+{IS}?        { printf("Constantes decimal \n"); }
{D}+{IS}?         { printf("Constante decimal\n"); }
L?'(\\.|[^\\'])+' { printf("Constante\n"); }

{D}+{E}{FS}?            { printf("Constantes notación científica\n"); }
{D}*"."{D}+({E})?{FS}?  { printf("Constantes notación científica\n"); }
{D}+"."{D}*({E})?{FS}?  { printf("Constantes notación científica\n"); }

L?\"(\\.|[^\\"])*\" { printf("Cadenas\n"); }

"..."         { printf("<Operador>\n"); }
">>="         { printf("<Asignación con desplazamiento a la derecha>\n"); }
"<<="         { printf("<Asignación con desplazamiento a la izquierda>\n"); }
```

Pruebas


Práctica04 — a.out ◂ make run — 53×17
luisrojas@MacBook-Air-de-Luis Práctica04 % make run
flex lexico.l
gcc -c lex.yy.c
gcc main.o lex.yy.o -ll
./a.out
while
<Ciclo>
Espacios
int
<Tipo de dato>
Espacios
&&
<Condición and>
Espacios
test
Tipo de dato
Espacios

Conclusiones

La práctica principalmente me ayudo a repasar la estructura del códgio Flex y su sintaxis, ya que por motivos técnicos tuve que reinstalar lo necesario e investigar sobre como usarlo. Además, en su mayoria pude comprender la especificación del lenguaje C que utilice, aunque aun me queda un poco de dudas sobre algunos aspectos, que espero comprender más adelante con los resltados. Una cosa que me di cuenta durante la programacación, es que aún hay cosas que desconozco del lenguaje C, como la función de algunas palabras reservadas u operadores, aunque afortunadamente encontre ejemplos en el libro que me ayudaron a entenderlas. Por último, aunque hasta ahora he encontrado cierto nivel de dificultad, espero poder conseguir buenos resultados con este compilador y en un futuro intentar hacer el proyecto inicial que propuse.

Referencias

Brian. W. Kernighan, Dennis M. Ritchie. (1991). El lenguaje de programación C. Unknow: Pearson Educación.

Manual de Flex, versión 2.5.35, febrero 2008.