



# **Instituto Politécnico Nacional**

## **Escuela Superior de Cómputo**

Alumno: Zapata Jasso José Rodolfo

Asignatura: Compiladores

Grupo: 3CV6

Docente: Saucedo Delgado Rafael Norman

Trabajo: Practica 4 Flex

## Introducción:

El lenguaje de entrada con el que se trabajará será C, mientras que el lenguaje de salida será ensamblador, se eligió este proyecto para aprobar la materia y conocer el mecanismo de dicho proceso, igualmente C se puede usar para trabajar con otras arquitecturas de manera sencilla.

### Lenguaje C

C es un lenguaje de programación creado en 1972 por Dennis M. Ritchie en los Laboratorios Bell como evolución del anterior lenguaje B, a su vez basado en BCPL. Al igual que B, es un lenguaje orientado a la implementación de Sistemas Operativos, concretamente Unix. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones.

Se trata de un lenguaje débilmente tipificado de medio nivel pero con muchas características de bajo nivel. Dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos.

La primera estandarización del lenguaje C fue en ANSI, con el estándar X3.159-1989. El lenguaje que define este estándar fue conocido vulgarmente como ANSI C. Posteriormente, en 1990, fue ratificado como estándar ISO (ISO/IEC 9899:1990). La adopción de este estándar es muy amplia por lo que, si los programas creados lo siguen, el código es portátil entre plataformas y/o arquitecturas.

### Lenguaje Ensamblador

El lenguaje ensamblador es el lenguaje de programación utilizado para escribir programas informáticos de bajo nivel, y constituye la representación más directa del Código máquina específico para cada arquitectura de computadoras legible por un programador. Aún hoy se utiliza en la programación de handler o manipuladores de dispositivos de hardware.

## 2. Desarrollo:

Se utilizará Flex versión tal para el desarrollo de la práctica.

### 2.1 Ejemplificar el lenguaje

#### 1: Estructura de un programa

Inclusión de bibliotecas, declaraciones de variables y tipos, y secuencia de funciones

Una y sólo una función denominada main (programa principal)

Función:

tipoRetorno Nombre (parametros) {sentencias}

*/\* ejemplo 1.- Escribe un mensaje en pantalla \*/*

*# include /\* incluye biblioteca donde se define E/S \*/*

*int main( )*

*{*

*/\*Este comentario es ignorado por el compilador y\*/*

*/\*no genera código \*/*

*printf("\n Introducción a la programación en lenguaje C");*

*return 0;*

*}*

#### 2: Definición de variables

Declaración variable: tipo Nombre [ =valor]

Asignación: variable = expresión

*/\* ejemplo 2.- multiplicar dos números enteros y muestra el resultado por pantalla \*/*

*#include*

*int main( )*

*{*

*int multiplicador; /\*se define multiplicador como un entero \*/*

*int multiplicando; /\*se define multiplicando como un entero \*/*

*int res; /\*se define resultado como un entero\*/*

*multiplicador = 1000; /\*se asignan valores\*/*

*multiplicando=2;*

*res=multiplicador\*multiplicando;*

*printf("Resultado =%d",res); /\*se muestra el resultado \*/*

*return 0;*

*}*

### **3: Definición de variables**

/\* ejemplo 3.- .- multiplica dos números enteros y muestra el resultado (utiliza definición múltiple de variables) \*/

```
#include
int main( )
{
    int multiplicador, multiplicando; /*se definen 2 variables*/
    multiplicador =1000; /*se les asigna valor*/
    multiplicando=2;
    printf("Resultado = %d", multiplicador*multiplicando);
    /*se muestra el resultado por pantalla*/
    return 0;
}
```

## **2.2 Clases léxicas, definición:**

Al realizar la práctica se identificaron las siguientes clases léxicas:

Almacenamiento  
Sentencia control  
Tipo de dato  
Compartir acceso  
Bucle  
Enumeración  
Almacenamiento  
Salto  
Devuelve  
Modificador de tipo de dato  
Operador  
Almacenamiento  
Estructura  
Asignar nombre  
Compartir espacio

## 2.3 Escribir las expresiones para cada clase

"auto"   "extern"   "register"   "static"	{printf("almacenamiento");}
"break"   "case"   "continue"   "default"     "else"   "if"   "switch"	{printf("sentenciacontrol");}
"char"   "double"   "float"   "int"   "long"   "void"	{printf("tipodato");}
"const"   "volatile"	{printf("compartiracceso");}
"do"   "for"   "while"	{printf("bucle");}
"enum"	{printf("enumeracion");}
"goto"	{printf("salto");}
"return"	{printf("devuelve");}
"short"   "signed"   "unsigned"	{printf("modificador tipodato");}
"sizeof"	{printf("operador");}
"struct"	{printf("estructura");}
"typedef"	{printf("asignar nombre");}
"union"	{printf("compartir espacio");}
{L}({L})({D})*	{printf("cero o varios valores L D");}
0[xX]{H}+{IS}?	{ printf("HEXADECIMAL"); }
0{D}+{IS}?	{ printf("UNSIGNED INT"); }
{D}+{IS}?	{ printf("LONG INT"); }
L?'(\\.[^\\'])*'	{ printf("CADENA"); }
{D}+{E}{FS}?	{ printf("LONG"); }
{D}*"."{D}+({E})?{FS}?	{ printf("FLOAT"); }
{D}+"."{D}*({E})?{FS}?	{ printf("DOUBLE"); }
"..."	{printf("operadorellipsis");}
">>="	{printf("asignacionderecha");}
"<<="	{printf("asignacionizquierda");}
"+="	{printf("operadorincremento");}
"-="	{printf("operadordecremento");}
"*="	{printf("operadormultiplicacion");}
"/="	{printf("operadordivision");}
"%="	{printf("operadormodulo");}
"&="	{printf("operadorlogico");}
"^="	{printf("operador");}
" ="	{printf("operadorlogico");}
">>"	{printf("operadoresplderecha");}
"<<"	{printf("operadoresplizquierda");}
"++"	{printf("operadorincremento");}
"--"	{printf("operadordecremento");}
"->"	{printf("apuntador");}
"&&"	{printf("operadory");}
"  "	{printf("operadoro");}
"<="	{printf("operadormenorigual");}

">="	{printf("operadormayorigual");}
"=="	{printf("operadorsiesigual");}
"!="	{printf("operadordiferente");}
","	{printf(" ");}
("{" "<%"	{printf(" {");}
("}" "%">")	{printf(" }");}
","	{printf(" ,");}
":"	{printf(" :");}
"="	{printf("operadorigual");}
"("	{printf(" (");}
")"	{printf(" )");}
("[" "<:"	{printf(" [");}
("]" "":>")	{printf(" ]");}
":"	{printf(" .");}
"&"	{printf("operador");}
"!"	{printf("operador");}
"~"	{printf("operador");}
"_"	{printf("operadorresta");}
"+"	{printf("operadorsuma");}
"*"	{printf("operador");}
"/"	{printf("operadordivision");}
"%"	{printf("operadormodulo");}
"<"	{printf("operadormayorque");}
">"	{printf("operadormenorque");}
"^"	{printf("operadorlogico");}
" "	{printf("operadorlogico");}
"?"	{printf("operadorlogico");}
%%	{printf("dividirbloques");}

## 2.4 Codificar en LEX (lexico.l)

Se incluye la codificación hecha en Flex, el nombre del archivo es: "léxico" formato ".l"

```
D          [0-9]
L          [a-zA-Z_]
H          [a-zA-F0-9]
E          [Ee][+]?{D}+
FS         (f|F|l|L)
IS         (u|U|l|L)*

%{
#include <stdio.h>
#include "y.tab.h"

void count();
}%

%%
"/**"      { comment(); }

"auto"     {printf("almacenamiento");}
"break"    {printf("sentenciacontrol");}
"case"     {printf("sentenciacontrol");}
"char"     {printf("tipodato");}
"const"    {printf("compartiracceso");}
"continue" {printf("sentenciacontrol");}
"default"  {printf("sentenciacontrol");}
"do"       {printf("bucle");}
"double"   {printf("tipodato");}
"else"     {printf("sentenciacontrol");}
"enum"     {printf("enumeracion");}
"extern"   {printf("almacenamiento");}
"float"    {printf("tipodato");}
"for"      {printf("bucle");}
"goto"     {printf("salto");}
"if"       {printf("sentenciacontrol");}
"int"      {printf("tipodato");}
"long"     {printf("tipodato");}
"register"  {printf("almacenamiento");}
"return"   {printf("devuelve");}
"short"    {printf("modificadortipodato");}
"signed"   {printf("modificadortipodato");}
"sizeof"   {printf("operador");}
"static"   {printf("almacenamiento");}
"struct"   {printf("estructura");}
```

"switch"	{printf("sentenciacontrol");}
"typedef"	{printf("asignarnombre");}
"union"	{printf("compartirespacio");}
"unsigned"	{printf("modificadortipodato");}
"void"	{printf("tipodato");}
"volatile"	{printf("compartiracceso");}
"while"	{printf("bucle");}
{L}({L}){D}*	{printf("cero o varios valored L D");}
0[xX]{H}+{IS}?	{ printf("HEXADECIMAL"); }
0{D}+{IS}?	{ printf("UNSIGNED INT"); }
{D}+{IS}?	{ printf("LONG INT"); }
L?'(\\.[^\\'])+	{ printf("CADENA"); }
{D}+{E}{FS}?	{ printf("LONG"); }
{D}*"."{D}+({E})?{FS}?	{ printf("FLOAT"); }
{D}+"."{D}*({E})?{FS}?	{ printf("DOUBLE"); }
L?'(\\.[^\\'])*\"	
"..."	{printf("operadorellipsis");}
">>="	{printf("asignacionderecha");}
"<<="	{printf("asignacionizquierda");}
"+="	{printf("operadorincremento");}
"-="	{printf("operadordecremento");}
"*="	{printf("operadormultiplicacion");}
"/="	{printf("operadordivision");}
"%="	{printf("operadormodulo");}
"&="	{printf("operadorlogico");}
"^="	{printf("operador");}
" ="	{printf("operadorlogico");}
">>"	{printf("operadoresplderecha");}
"<<"	{printf("operadoresplizquierda");}
"++"	{printf("operadorincremento");}
"--"	{printf("operadordecremento");}
"->"	{printf("apuntador");}
"&&"	{printf("operadory");}
"  "	{printf("operadoro");}
"<="	{printf("operadormenorigual");}
">="	{printf("operadormayorigual");}
"=="	{printf("operadorsiesigual");}
"!="	{printf("operadordiferente");}
","	{printf(" ,"); }
("{" "<%"")	{printf(" {"); }
(")" "%">")	{printf(" }");}
","	{printf(" ,");}
":"	{printf(" :");}



```

"="                {printf("operadorigual");}
"("                {printf(" (");}
")"                {printf(" ");}
("[|" "<:")        {printf(" [|");}
("]" "|":>")       {printf(" ]");}
"."                {printf(" .");}
"&"                {printf("operador");}
"!"                {printf("operador");}
"~"                {printf("operador");}
"_"                {printf("operadorresta");}
"+"                {printf("operadorsuma");}
"*"                {printf("operador");}
"/"                {printf("operadordivision");}
%"                {printf("operadormodulo");}
"<"                {printf("operadormayorque");}
">"                {printf("operadormenorque");}
"^"                {printf("operadorlogico");}
"|"                {printf("operadorlogico");}
"?"                {printf("operadorlogico");}

```

```
[ \t\n\r]
```

```
%%                {printf("dividirbloques");}
```

Se incluyen 2 códigos, los cuales son main.c y makefile.

```

change.log x makefile.jpg x main.c x
1  lex.yy.c: lexico.l
2      flex lexico.l
3  main.o: main.c
4      gcc -c main.c
5
6  lex.yy.o: lex.yy.c
7      gcc -c lex.yy.c
8
9  a.out: main.o lex.yy.o
10     gcc main.o lex.yy.o -lfl
11
12 clean:
13     rm -f a.out main.o lex.yy.o lex.yy.c
14
15 run: a.out
16     ./a.out

```

```

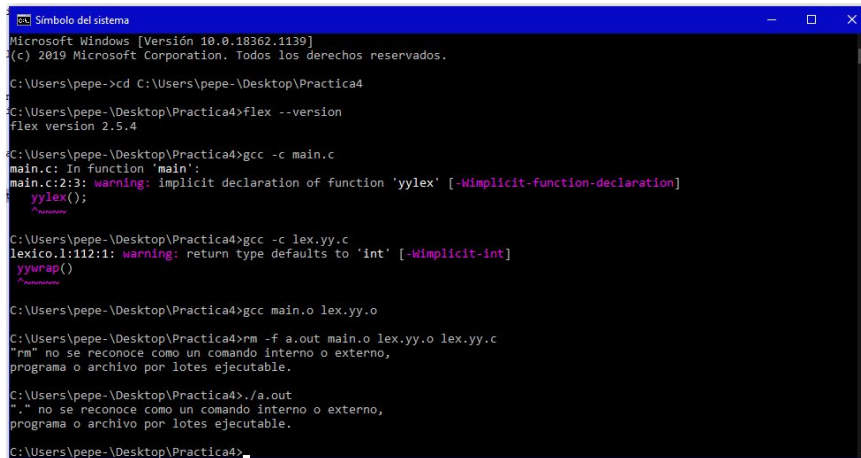
change.log x makefile.jpg x main.c x
1  int main(void) {
2      yylex();
3      return 0;
4  }
5

```

## 2.5 Pruebas (compilar, introducir, programas)

Aquí se muestra la terminal en windows con distintas entradas.

El archivo Makefile se usa para compilar archivos y luego se utiliza make clean para borrar..



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.18362.1139]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\pepe->cd C:\Users\pepe\Desktop\Practica4
C:\Users\pepe\Desktop\Practica4>flex --version
flex version 2.5.4

C:\Users\pepe\Desktop\Practica4>gcc -c main.c
main.c: In function 'main':
main.c:2:3: warning: implicit declaration of function 'yylex' [-Wimplicit-function-declaration]
  yylex();
  ^~~~~~

C:\Users\pepe\Desktop\Practica4>gcc -c lex.yy.c
lexico.l:112:1: warning: return type defaults to 'int' [-Wimplicit-int]
  yywrap()
  ^~~~~~

C:\Users\pepe\Desktop\Practica4>gcc main.o lex.yy.o
C:\Users\pepe\Desktop\Practica4>rm -f a.out main.o lex.yy.o lex.yy.c
"rm" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

C:\Users\pepe\Desktop\Practica4>./a.out
",." no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

C:\Users\pepe\Desktop\Practica4>
```

## Conclusiones:

El desarrollo de la práctica fue interesante y en lo personal bastante complejo, el conocimiento de AFD y AFN es fundamental para realizar esta práctica, también se podría decir que cada programador elige cómo agrupar las clases léxicas.

Por último tuve muchos problemas en mi equipo para instalar máquinas virtuales, y particiones, solo me quedó la opción de usar windows y por ello no termine todo bien.

**Referencias:**       Manual de flex.

