

Flex y Bison - capitulo 1

Taller 2

Jonathan David Valdes Gonzalez

Joaquin Fernando Sanchez Cifuentes

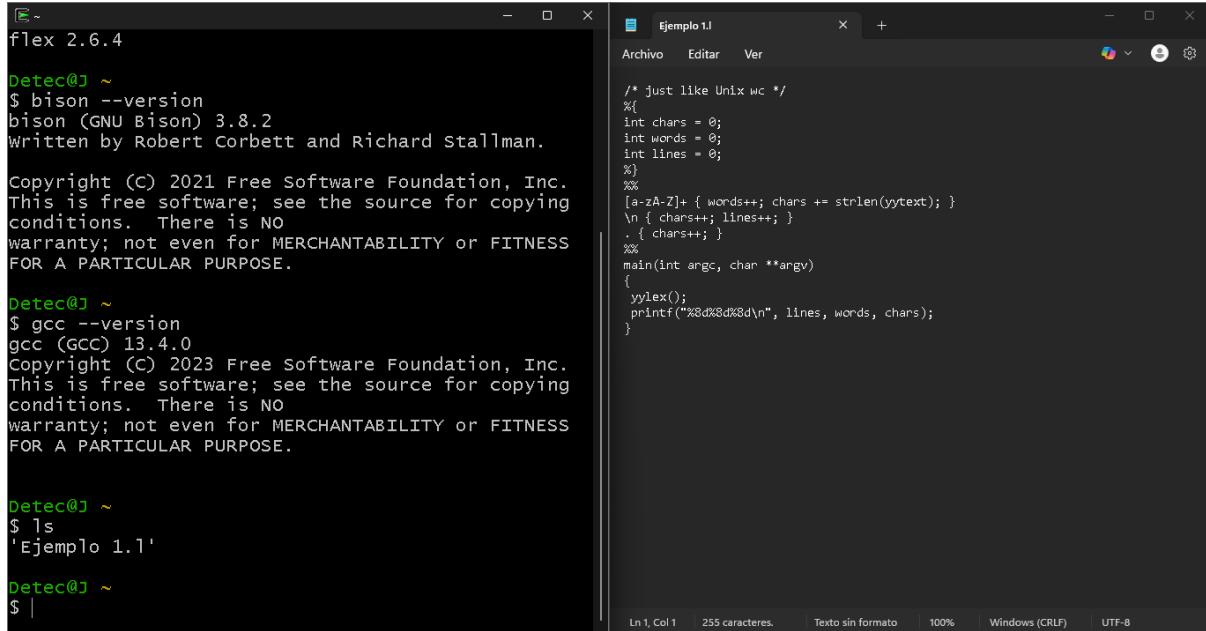
Universidad Sergio Arboleda

Ciencias de la computación e Inteligencia Artificial

Lenguaje de programación y transducción

No me odies profe, la próxima si ejecuto todo esto en un sistema operativo basado en Linux 😞 .

## Ejemplo 1 (Lineas, Palabras y Caracteres):



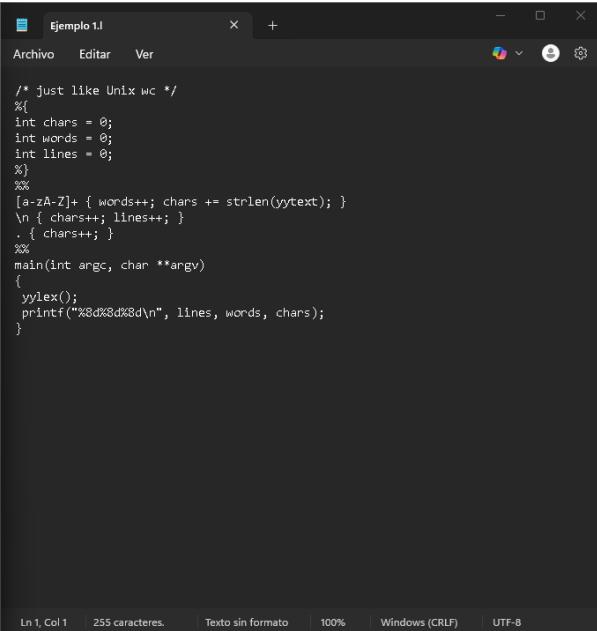
```
flex 2.6.4
Detec@J ~
$ bison --version
bison (GNU Bison) 3.8.2
Written by Robert Corbett and Richard Stallman.

Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying
conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS
FOR A PARTICULAR PURPOSE.

Detec@J ~
$ gcc --version
gcc (GCC) 13.4.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying
conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS
FOR A PARTICULAR PURPOSE.

Detec@J ~
$ ls
'Ejemplo 1.1'

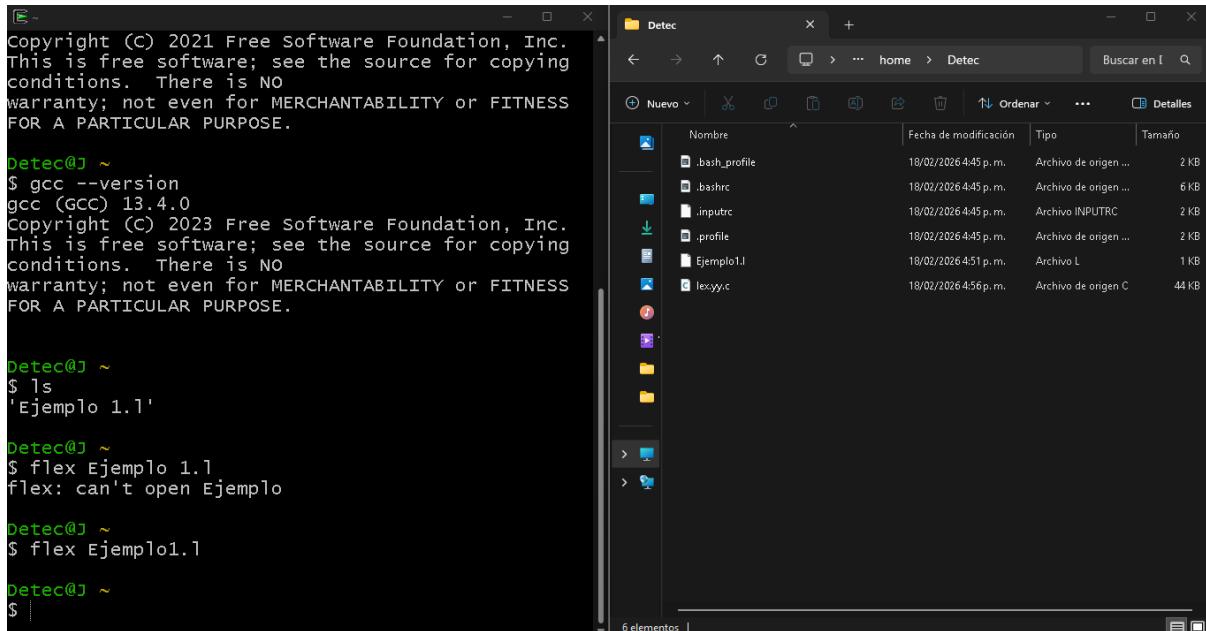
Detec@J ~
$ |
```



```
/* just like Unix wc */
{
int chars = 0;
int words = 0;
int lines = 0;
}
*/
[a-zA-Z]+ { words++; chars += strlen(yytext); }
\n { chars++; lines++; }
. { chars++; }
}
main(argc, char **argv)
{
yylex();
printf("%d%d%d\n", lines, words, chars);
}
```

Ln 1, Col 1 255 caracteres. Texto sin formato 100% Windows (CRLF) UTF-8

1. Lee el archivo de reglas Ejemplo1.l (le quite el espacio para que funcione)
2. Se crea el archivo C lex.yy.c con la función yylex().



```
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying
conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS
FOR A PARTICULAR PURPOSE.

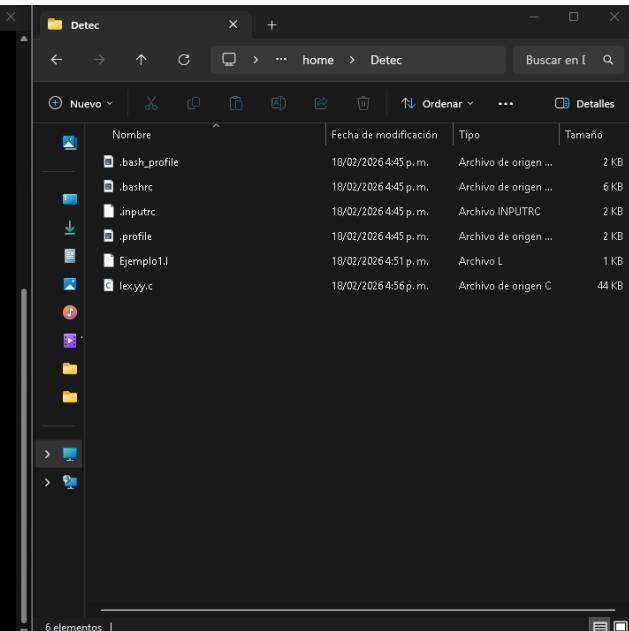
Detec@J ~
$ gcc --version
gcc (GCC) 13.4.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying
conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS
FOR A PARTICULAR PURPOSE.

Detec@J ~
$ ls
'Ejemplo 1.1'

Detec@J ~
$ flex Ejemplo 1.1
flex: can't open Ejemplo

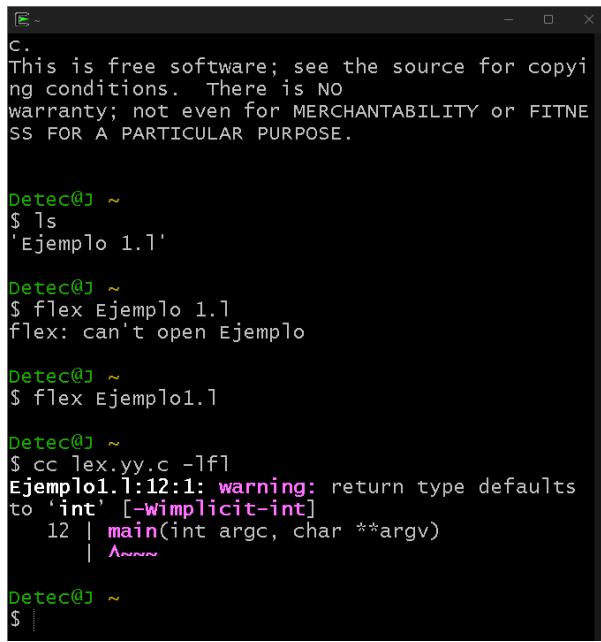
Detec@J ~
$ flex Ejemplo1.l

Detec@J ~
$ |
```



Nombre	Fecha de modificación	Tipo	Tamaño
.bash_profile	18/02/2026 4:45 p.m.	Archivo de origen ...	2 KB
.bashrc	18/02/2026 4:45 p.m.	Archivo de origen ...	6 KB
.inputrc	18/02/2026 4:45 p.m.	Archivo INPUTRC	2 KB
.profile	18/02/2026 4:45 p.m.	Archivo de origen ...	2 KB
Ejemplo1.l	18/02/2026 4:51 p.m.	Archivo L	1 KB
lex.yy.c	18/02/2026 4:56 p.m.	Archivo de origen C	44 KB

3. Compila el C generado. -lfl enlaza la biblioteca de flex.  
Crea el ejecutable (por defecto a.out; esta vez se crea un a.exe).



This terminal session shows the creation of a lexical analyzer (lex) and its execution:

```

Detec@J ~
$ ls
'Ejemplo 1.l'

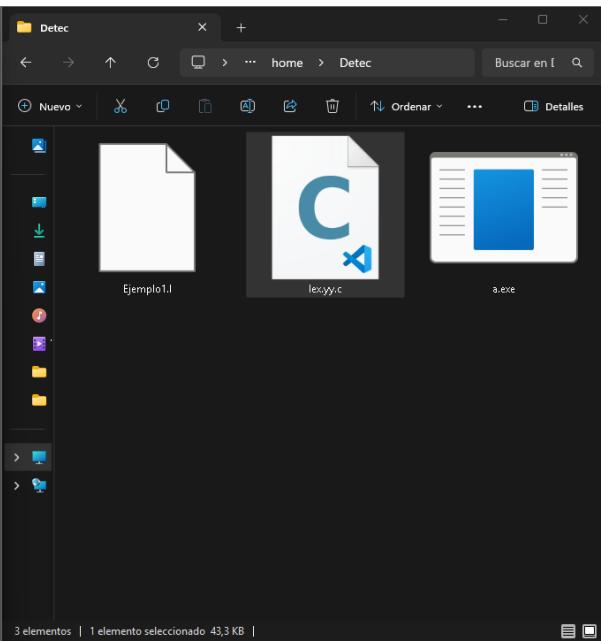
Detec@J ~
$ flex Ejemplo 1.l
flex: can't open Ejemplo

Detec@J ~
$ flex Ejemplo1.l

Detec@J ~
$ cc lex.yy.c -lfl
Ejemplo1.l:12:1: warning: return type defaults
to 'int' [-Wimplicit-int]
  12 | main(int argc, char **argv)
     | ^~~~

Detec@J ~
$ 

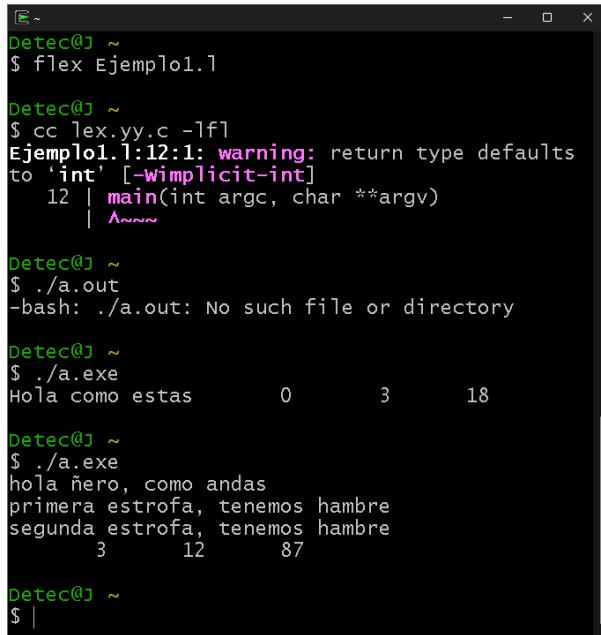
```



The file explorer shows three files in the 'Detc' folder:

- Ejemplo1.l
- lex.yy.c
- a.exe

#### 4. Ejecutamos a.exe y probamos el código.



This terminal session shows the execution of the generated executable (a.exe) and its output:

```

Detec@J ~
$ flex Ejemplo1.l

Detec@J ~
$ cc lex.yy.c -lfl
Ejemplo1.l:12:1: warning: return type defaults
to 'int' [-Wimplicit-int]
  12 | main(int argc, char **argv)
     | ^~~~

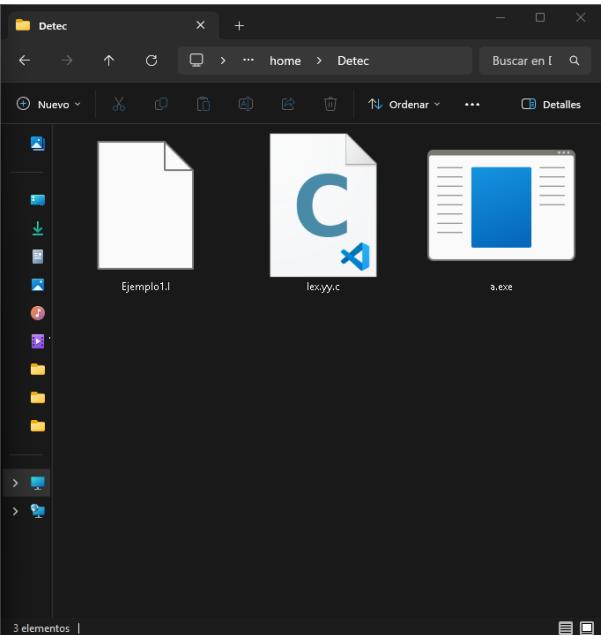
Detec@J ~
$ ./a.out
./a.out: No such file or directory

Detec@J ~
$ ./a.exe
Hola como estas      0      3      18

Detec@J ~
$ ./a.exe
hola ñero, como andas
primera estrofa, tenemos hambre
segunda estrofa, tenemos hambre
  3      12      87

Detec@J ~
$ 

```



The file explorer shows three files in the 'Detc' folder:

- Ejemplo1.l
- lex.yy.c
- a.exe

Ejemplo 2 (Traductor):

```

$ cc lex.yy.c -lfl
Ejemplo1.l:12:1: warning: return type defaults to 'int' [-Wimplicit-int]
  12 | main(int argc, char **argv)
     | ^~~~

Detec@J ~
$ ./a.out
-bash: ./a.out: No such file or directory

Detec@J ~
$ ./a.exe
Hola como estas      0      3      18

Detec@J ~
$ ./a.exe
hola ñero, como andas
primera estrofa, tenemos hambre
segunda estrofa, tenemos hambre
  3      12      87

Detec@J ~
$ ls
Ejemplo1.l  Ejemplo2.l  a.exe  lex.yy.c

Detec@J ~
$ flex Ejemplo2.l

```

Ln 10, Col 3 244 caracteres. Texto sin formato 100% Windows (CRLF) UTF-8

Cambie los nombres de los archivos para que no se repitan los nombres en los archivos y se dividan bien los ejecutables.

```

Detec@J ~
$ flex Ejemplo2.l
Ejemplo2.l:1: premature EOF

Detec@J ~
$ flex Ejemplo2.l
Ejemplo2.l:1: premature EOF

Detec@J ~
$ ^C

Detec@J ~
$ flex Ejemplo2.l

Detec@J ~
$ flex Ejemplo1.l

Detec@J ~
$ cc Ejemplo1.yy.c -lfl
Ejemplo1.l:12:1: warning: return type defaults to 'int' [-Wimplicit-int]
  12 | main(int argc, char **argv)
     | ^~~~

Detec@J ~
$ cc Ejemplo2.yy.c -lfl

```

Dirección: home > Detec

Ejemplo1.exe Ejemplo1.l Ejemplo1yy.c

Ejemplo2.l Ejemplo2.yy.c

5 elementos

- Ejecutamos el traductor

```

Detec@J ~
$ flex Ejemplo2.l

Detec@J ~
$ flex Ejemplo1.l

Detec@J ~
$ cc Ejemplo1.yy.c -lfl
Ejemplo1.l:12:1: warning: return type defaults
to 'int' [-Wimplicit-int]
 12 | main(int argc, char **argv)
     | ^~~~

Detec@J ~
$ cc Ejemplo2.yy.c -lfl

Detec@J ~
$ ./Ejemplo2.exe
colour
color
flavour
flavor
clever
smart

```

### Ejemplo 3 (Calculadora):

```

Detec@J ~
$ cc Ejemplo1.yy.c -lfl
Ejemplo1.l:12:1: warning: return type defaults
to 'int' [-Wimplicit-int]
 12 | main(int argc, char **argv)
     | ^~~~

Detec@J ~
$ cc Ejemplo2.yy.c -lfl

Detec@J ~
$ ./Ejemplo2.exe
colour
color
flavour
flavor
clever
smart

Detec@J ~
$ ls
Ejemplo1.exe  Ejemplo2.exe  Ejemplo3.l
Ejemplo1.l    Ejemplo2.l    Ejemplo3.l
Ejemplo1.yy.c Ejemplo2.yy.c Ejemplo3.l

Detec@J ~
$ 

```

### 1. Creamos el escáner

```

E~
to 'int' [-Wimplicit-int]
12 | main(int argc, char **argv)
| ^~~~

Detec@J ~
$ cc Ejemplo2.yy.c -lfl

Detec@J ~
$ ./Ejemplo2.exe
colour
color
flavour
flavor
clever
smart

Detec@J ~
$ ls
Ejemplo1.exe    Ejemplo2.exe    Ejemplo3.l
Ejemplo1.l      Ejemplo2.l      Ejemplo1.yy.c
Ejemplo1.yy.c   Ejemplo2.yy.c

Detec@J ~
$ flex Ejemplo3.l

Detec@J ~
$ |

```

## 2. Compilamos el C en un ejecutable

```

E~
Detec@J ~
$ cc Ejemplo2.yy.c -lfl

Detec@J ~
$ ./Ejemplo2.exe
colour
color
flavour
flavor
clever
smart

Detec@J ~
$ ls
Ejemplo1.exe    Ejemplo2.exe    Ejemplo3.l
Ejemplo1.l      Ejemplo2.l      Ejemplo1.yy.c
Ejemplo1.yy.c   Ejemplo2.yy.c

Detec@J ~
$ flex Ejemplo3.l

Detec@J ~
$ cc Ejemplo3.yy.c -lfl

Detec@J ~
$ |

```

## 3. Ejecutamos y probamos.

The terminal window shows the following session:

```
Ejemplo1.yy.c Ejemplo2.yy.c
Detec@J ~
$ flex Ejemplo3.l

Detec@J ~
$ cc Ejemplo3.yy.c -lfl

Detec@J ~
$ ./Ejemplo3.exe
5*32
NUMBER 5
TIMES
NUMBER 32
NEWLINE
5+5
NUMBER 5
PLUS
NUMBER 5
NEWLINE
xd
Mystery character x
Mystery character d
NEWLINE

Detec@J ~
```

The file explorer window shows the following files in the 'Detec' folder:

- Ejemplo1.exe
- Ejemplo1.l
- Ejemplo1.yy.c
- Ejemplo2.exe
- Ejemplo2.l
- Ejemplo2.yy.c
- Ejemplo3.l
- Ejemplo3.yy.c
- Ejemplo3.exe

#### Ejemplo 4 (Reconocer tokens de una calculadora):

The terminal window shows the following session:

```
Ejemplo1.yy.c Ejemplo2.yy.c
Detec@J ~
$ flex Ejemplo3.l

Detec@J ~
$ cc Ejemplo3.yy.c -lfl

Detec@J ~
$ ./Ejemplo3.exe
5*32
NUMBER 5
TIMES
NUMBER 32
NEWLINE
5+5
NUMBER 5
PLUS
NUMBER 5
NEWLINE
xd
Mystery character x
Mystery character d
NEWLINE

Detec@J ~
$ flex Ejemplo4.l
```

The code editor window displays the generated C code for 'Ejemplo4.l':

```
%{
enum yytokentype {
    NUMBER = 258,
    ADD = 259,
    SUB = 260,
    MUL = 261,
    DIV = 262,
    ABS = 263,
    EOL = 264
};

int yyval;
%}
%%
"+" { return ADD; }
"-" { return SUB; }
"*" { return MUL; }
"/" { return DIV; }
"|" { return ABS; }
[0-9]+ { yyval = atoi(yytext); return NUMBER; }
\n { return EOL; }
[ \t] { /* ignore whitespace */ }
. { printf("Mystery character %c\n", *yytext); }
%%
main(int argc, char **argv)
{
    int tok;
    while(tok = yylex()) {
        printf("%d", tok);
        if(tok == NUMBER) printf(" = %d\n", yyval);
        else printf("\n");
    }
}
```

Saltamos los pasos anteriores y ejecutamos el código.

Detect@J ~

```
$ ./Ejemplo4.exe
a /34 + |45
Mystery character a
262
258 = 34
259
263
258 = 45
264
34+34
258 = 34
259
258 = 34
264
34*5
258 = 34
261
258 = 5
264
holaaaaaa
Mystery character h
Mystery character o
Mystery character l
Mystery character a
Mystery character a
Mystery character a
```

File Explorer content:

- Ejemplo2.exe
- Ejemplo2.l
- Ejemplo2.yyc
- Ejemplo3.exe
- Ejemplo3.l
- Ejemplo3.yyc
- Ejemplo4.l
- Ejemplo4.yyc
- Ejemplo4.exe

Ejemplo 5 y 6 (Calculadora simple): Este ejemplo debe utilizar flex y bison para su funcionamiento.

## 1. Código en flex:

```
.c:(.rdata$.refptr.yylval[.refptr.yylval]+0x0):
undefined reference to `yylval'
collect2: error: ld returned 1 exit status
```

Detect@J ~

```
$ bison -d Ejemplo5bison.y
```

Detect@J ~

```
$ flex Ejemplo5flex.l
```

Detect@J ~

```
$ cc -o $@ Ejemplo5bison.tab.c flex5.yy.c -lfl
cc1: fatal error: flex5.yy.c: No such file or d
irectory
compilation terminated.
```

Detect@J ~

```
$ cc -o $@ Ejemplo5bison.tab.c flex5.yy.c -lfl
/usr/lib/gcc/x86_64-pc-cygwin/13/../../../../x8
6_64-pc-cygwin/bin/ld: /tmp/cc5ACbzD.o:flex5.yy
.c:(.rdata$.refptr.yylval[.refptr.yylval]+0x0):
undefined reference to `yylval'
collect2: error: ld returned 1 exit status
```

Detect@J ~

```
$ flex Ejemplo5flex.l
```

Code Editor content (flex5.l):

```
%{
#include "Ejemplo5bison.tab.h"
#include <stdlib.h>
%}

"+"
"-"
"*"
"/"
"|"
[0-9]+
\n
[ \t]+
.

%{
```

## 2. Código en bison

```

.c:(.rdata$._refptr.yylval[.refptr.yylval]+0x0):
 undefined reference to `yylval'
collect2: error: ld returned 1 exit status

Detec@J ~
$ bison -d Ejemplo5bison.y

Detec@J ~
$ flex Ejemplo5flex.l

Detec@J ~
$ cc -o $@ Ejemplo5bison.tab.c flex5.yy.c -lfl
cc1: fatal error: flex5.yy.c: No such file or directory
compilation terminated.

Detec@J ~
$ cc -o $@ Ejemplo5bison.tab.c flex5.yy.c -lfl
/usr/lib/gcc/x86_64-pc-cygwin/13/../../../../x86_64-pc-cygwin/bin/ld: /tmp/cc5ACbzD.o:flex5.yy.c:(.rdata$._refptr.yylval[.refptr.yylval]+0x0):
 undefined reference to `yylval'
collect2: error: ld returned 1 exit status

Detec@J ~
$ bison -d Ejemplo5bison.y

```

Code Editor Content:

```

%include <stdio.h>
%include <stdlib.h>

int yylex(void);
void yyerror(const char *s);

#define YYSTYPE int

%token NUMBER
%token ADD SUB MUL DIV ABS
%token EOL
%%

callist:
/* vacio */
| callist exp EOL { printf("%d\n", $2); }

exp:
Factor
| exp ADD Factor { $$. = $1 + $3; }
| exp SUB Factor { $$. = $1 - $3; }

Factor:
term
| Factor MUL term { $$. = $1 * $3; }
| Factor DIV term { $$. = $1 / $3; }

term:
NUMBER
| ADD term { $$ = ($2 >> 6) ? $2 : -$2; }

%%

int main(void)
{
    return yyparse();
}

void yyerror(const char *s)
{
    fprintf(stderr, "error: %s\n", s);
}

```

Terminal Status Bar:

Ln 12, Col 27 662 caracteres. Texto sin formato 70% Windows (CRLF) UTF-8

### 3. Ahora compilamos, ejecutamos y hacemos las pruebas

```

$ bison -d Ejemplo5bison.y

Detec@J ~
$ flex Ejemplo5flex.l

Detec@J ~
$ gcc -o $ Ejemplo5bison.tab.c lex.yy.c -lfl

Detec@J ~
$ ./$.exe
5*5
= 25
3/2
= 1
6+6
= 12
6+530
= 536
50/5
= 10
15/15
= 1
holá
Caracter desconocido: h
Caracter desconocido: o
Caracter desconocido: l
Caracter desconocido: a
error: syntax error

Detec@J ~
$ |

```

File Explorer View:

- Ejecución: Ejemplo5.exe
- Fuentes: Ejemplo5.i, Ejemplo3.i, Ejemplo3yy.c
- Archivos temporales: Ejemplo4.exe, Ejemplo4.i, Ejemplo4yy.c
- Archivos generados: Ejemplo5bison.y, Ejemplo5flex.l, Ejemplo5bison.tab.h, Ejemplo5bison.tab.c, lex.yy.c, Ejemplo5bison.tab.o, Ejemplo5bison.o, Ejemplo5bison\$exe

## Ejercicios

1. ¿Aceptará la calculadora una línea que solo contenga un comentario? ¿Por qué no? ¿Sería más fácil solucionar esto en el escáner o en el analizador?

No, normalmente no la acepta como una entrada válida. el escáner ignora comentarios (no retorna token). En el parser es más fácil arreglarlo al agregar una regla que acepte una línea vacía.

**2. Convierta la calculadora en una calculadora hexadecimal que acepte números hexadecimales y decimales.**

## - Archivo bison

```
Ejercicio2flex.1:11:10: error: 'DIV' undeclared (first use in this function)
  11 |     "/"           { return DIV; }
      |             ^~~~

Ejercicio2flex.1:16:14: error: incompatible types when assigning to type 'YYSTYPE' from type 'long int'
  16 |     yylval = strtol(yytext, NULL, 16
);
      |             ^~~~~~

Ejercicio2flex.1:21:14: error: incompatible types when assigning to type 'YYSTYPE' from type 'long int'
  21 |     yylval = strtol(yytext, NULL, 10
);
      |             ^~~~~~

Detec@J ~
$ ^C

Detec@J ~
$ bison -d Ejercicio2bison.y

Detec@J ~
```

#### - Archivo flex

```
11 |     "/"      { return DIV; }
|     ^~~~  
Ejercicio2flex.l:16:14: error: incompatible
types when assigning to type 'YYSTYPE' from
type 'long int'
16 |     yylval = strtol(yytext, NULL, 16
);  
|     ^~~~~~  
Ejercicio2flex.l:21:14: error: incompatible
types when assigning to type 'YYSTYPE' from
type 'long int'
21 |     yylval = strtol(yytext, NULL, 10
);  
|     ^~~~~~  
  
Detec@J ~  
$ ^C  
  
Detec@J ~  
$ bison -d Ejercicio2bison.y  
  
Detec@J ~  
$ flex Ejercicio2flex.l  
  
Detec@J ~
```

```
11 |     "/"      { return DIV; }
|     ^~~~  
Ejercicio2flex.l:16:14: error: incompatible
types when assigning to type 'YYSTYPE' from
type 'long int'
16 |     yylval = strtol(yytext, NULL, 16
);  
|     ^~~~~~  
Ejercicio2flex.l:21:14: error: incompatible
types when assigning to type 'YYSTYPE' from
type 'long int'
21 |     yylval = strtol(yytext, NULL, 10
);  
|     ^~~~~~  
  
Detec@J ~  
$ ^C  
  
Detec@J ~  
$ bison -d Ejercicio2bison.y  
  
Detec@J ~  
$ flex Ejercicio2flex.l  
  
Detec@J ~
```

```
%{
#include "Ejercicio2bison.tab.h"
#include <stdlib.h>
%}

%%

"+"
{ return '+'; }

"-"
{ return '-'; }

"*"
{ return '*' ; }

"/"
{ return '/' ; }

"|" { return ABS; }

"&" { return AND; }

"||" { return OR; }

0x[0-9a-fA-F]+ {
    yylval.val = strtol(yytext, NULL, 16);
    return NUMBER;
}

[0-9]+ {
    yylval.val = strtol(yytext, NULL, 10);
    return NUMBER;
}

\n { return EOL; }

[ \t]+ ;
.

{ printf("Carácter desconocido: %s\n", yytext); }

%%
```

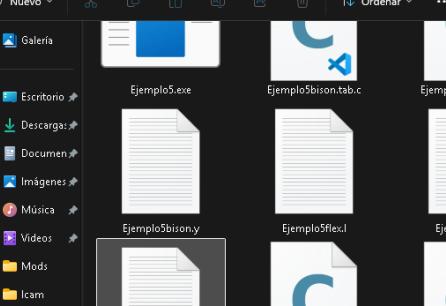
#### - Ejecución del código

```
Detec@J ~
$ bison -d Ejercicio2bison.y

Detec@J ~
$ flex Ejercicio2flex.l

Detec@J ~
$ gcc -o $ Ejercicio2bison.tab.c lex.yy.c -lfl

Detec@J ~
$ ./$.exe
5+5
= 10 (0xA)
5*5
= 25 (0x19)
15+15
= 30 (0x1E)
2*30
= 60 (0x3C)
0x1e
= 30 (0x1E)
0xff
= 255 (0xFF)


```

### 3. Agregue operadores de bits como AND y OR a la calculadora.

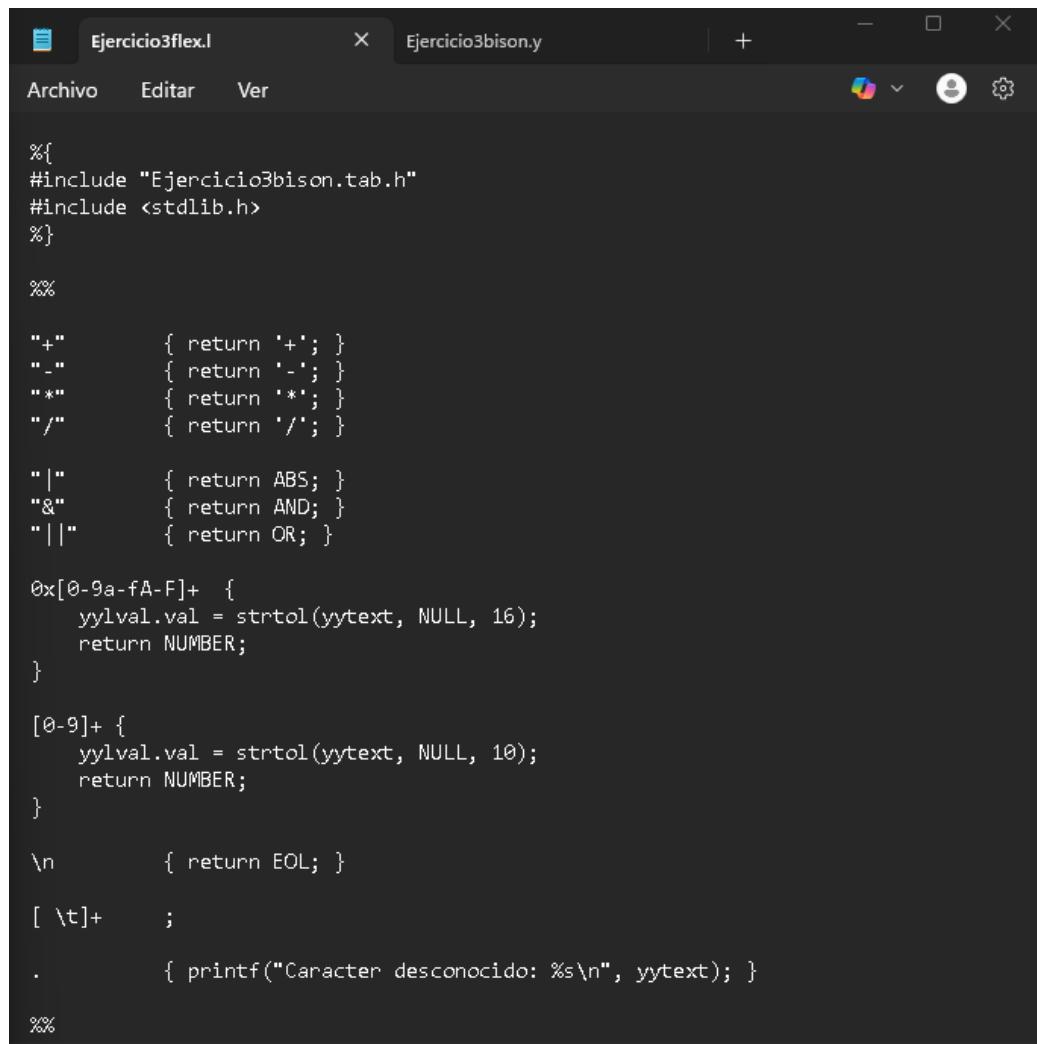
The terminal window displays the following text:

```
aracter desconocido: [  
Caracter desconocido: A  
Caracter desconocido:  
aracter desconocido: [  
Caracter desconocido: B  
Caracter desconocido:  
aracter desconocido: [  
Caracter desconocido: C  
Caracter desconocido:  
aracter desconocido: [  
Caracter desconocido: D  
Caracter desconocido:  
= 14 (0xE)  
10|14  
= 14 (0xE)  
  
Detec@J ~  
$ ./$.exe  
10+10  
= 20 (0x14)  
10|14  
= 14 (0xE)  
(10+5)&7  
Caracter desconocido: (  
Caracter desconocido: )  
= 7 (0x7)  
0xf&0xa  
= 10 (0xA)
```

The file explorer window shows the following files in the 'Detec' folder:

- Ejercicio2bison.tab.c
- Ejercicio2bison.tab.h
- Ejercicio2bison.y
- Ejercicio2flex.l
- Ejercicio2flex.y
- Ejercicio3bison.tab.c
- Ejercicio3bison.tab.h
- Ejercicio3flex.l
- Ejercicio3flex.y
- Ejercicio3bison.y
- flexejeryy.c
- lexyy.c
- \$exe

Details pane: 30 elementos | 1 elemento seleccionado 1,13 KB |



The screenshot shows a code editor window with a dark theme. The title bar displays two tabs: "Ejercicio3flex.l" and "Ejercicio3bison.y". The main area contains the following Flex lexer grammar code:

```
%{  
#include "Ejercicio3bison.tab.h"  
#include <stdlib.h>  
%}  
  
%%  
  
"+"      { return '+'; }  
"-"      { return '-'; }  
"**"     { return '*' ; }  
"/"      { return '/' ; }  
  
"|"      { return ABS; }  
"&"     { return AND; }  
"||"     { return OR; }  
  
0x[0-9a-fA-F]+ {  
    yylval.val = strtol(yytext, NULL, 16);  
    return NUMBER;  
}  
  
[0-9]+ {  
    yylval.val = strtol(yytext, NULL, 10);  
    return NUMBER;  
}  
  
\n      { return EOL; }  
  
[ \t]+ ;  
.       { printf("Caracter desconocido: %s\n", yytext); }  
  
%%
```

```

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

int yylex(void);
void yyerror(const char *s);
%}

/* tipo semántico */
%union {
    long val;
}

/* tokens */
%token AND OR
%token <val> NUMBER
%token EOL
%token ABS

/* precedencias */
%left OR
%left AND
%left '+' '-'
%left '*' '/'
%right UMINUS

%type <val> expr
%%

input:
    /* vacío */
    | input_line
    ;

line:
    EOL
    | expr EOL { printf("= %ld (0x%lx)\n", $1, $1); }
    ;

expr:
    NUMBER          { $$ = $1; }
    | expr '+' expr { $$ = $1 + $3; }
    | expr '-' expr { $$ = $1 - $3; }
    | expr '*' expr { $$ = $1 * $3; }
    | expr '/' expr { $$ = $1 / $3; }
    | '-' expr %prec UMINUS { $$ = -$2; }

    | ABS expr ABS   { $$ = ($2 >= 0) ? $2 : -$2; }
    | expr AND expr  { $$ = $1 & $3; }
    | expr OR  expr   { $$ = $1 | $3; }
    | '(' expr ')'   { $$ = $2; }
    ;

void yyerror(const char *)

```

#### 4. ¿La versión manuscrita del escáner del Ejemplo 4 reconoce exactamente los mismos tokens que la versión flex?

Si el código manual replica con precisión:

- las mismas expresiones regulares,
- el mismo orden de prioridad,
- y el mismo manejo de errores,

entonces sí podrían coincidir.

En la práctica, normalmente **no reconocen exactamente el mismo conjunto de tokens en todos los casos límite**.

## 5. ¿Se te ocurren lenguajes para los que Flex no sería una buena herramienta para escribir un escáner?

Flex falla cuando el léxico depende de:

- Anidación arbitraria
- Contexto semántico
- Retroalimentación del parser

## 6. Reescribe el programa de conteo de palabras en C. Ejecuta archivos grandes en ambas versiones.

The screenshot shows two windows side-by-side. The left window is a terminal window titled 'Detec@J ~' displaying the output of a Flex-generated scanner. It shows various characters and their hex values, followed by a section of code from 'ArchivoC\_Ejercicio6.c'. The right window is a code editor titled 'ArchivoC\_Ejercicio6.c' containing the C source code for a word counter. The terminal output shows the command 'gcc -o ArchivoC\_Ejercicio6 ArchivoC\_Ejercicio6.c' being run, followed by the execution of the program with the input 'hola como estas que locooo', which outputs '2 5 27'.

```
Carácter desconocido:  
= 14 (0xE)  
10||4  
= 14 (0xE)  
  
Detec@J ~  
$ ./$.exe  
10+10  
= 20 (0x14)  
10||4  
= 14 (0xE)  
(10+5)&7  
Carácter desconocido: (  
Carácter desconocido: )  
= 7 (0x7)  
0xf&0xa  
= 10 (0xA)  
  
Detec@J ~  
$ gcc -o ArchivoC_Ejercicio6 ArchivoC_Ejercicio6.c  
Detec@J ~  
$ ./ArchivoC_Ejercicio6.exe  
hola como estas  
que locooo  
2 5 27  
  
Detec@J ~  
$ |
```

```
#include <stdio.h>  
#include <ctype.h>  
  
int main(void)  
{  
    int c, linword = 0;  
    long chars = 0, words = 0, lines = 0;  
  
    while ((c = getchar()) != EOF) {  
        chars++;  
  
        if (c == '\n')  
            lines++;  
  
        if (isspace(c))  
            linword = 0;  
        else if (!linword) {  
            linword = 1;  
            words++;  
        }  
    }  
  
    printf("%ld%ld%ld\n", lines, words, chars);  
    return 0;  
}
```

Ln 1, Col 1 427 caracteres. Texto sin formato 100% Windows (CRLF) UTF-8