

Grupo Jueves 12:00 – 14:00 semanas B
- Práctica 3 -
Autor : Rubén Rodríguez Esteban

Ejercicio 1:

1. Descripción del ejercicio:

Observa el contenido de los ficheros `calcOrig.l` y `calcOrig.y` ¿Qué mensajes has recibido de bison? El objetivo de estos fuentes es implementar una calculadora sencilla de enteros positivos, con las operaciones de suma, resta, multiplicación y división con la precedencia habitual.

Prueba esta calculadora con distintas expresiones aritméticas, ¿para cuáles da error? Observa el código alternativo en `calcMejor.l` y `calcMejor.y` . ¿Qué diferencias observas? ¿Cómo funciona en los casos que fallaban antes? ¿Por qué?

1.1 Respuesta a las cuestiones:

Al compilar y ejecutar los ficheros dados, no existen fallos de compilación. Sin embargo, si existen advertencias o warnings en determinados puntos del programa, concretamente en `yyerror` y en `yychar`. Esto se debe a que en estos ficheros, debajo de la biblioteca `<stdio.h>` es conveniente poner las siguientes líneas de código:

- `int yylex();`
- `int yyerror();`

Así, declaramos estos dos campos, que inicialmente de acuerdo con los warnings, no habían sido declarados.

1.2 Primera versión de la calculadora:

Si se compila los ficheros `calcOrig.l` y `calcOrig.y` y se ejecuta el archivo resultante se pueden observar los siguientes sucesos:

- El programa realiza bien operaciones tales como sumas y restas del tipo separando los caracteres con espacios. También es capaz de realizar operaciones con números enteros. Las multiplicaciones y divisiones no presenta ningún problema. Además también puede realizar todas las operaciones anteriores entre paréntesis. Sin embargo, no sabe operar si los caracteres están juntos, y tampoco sabe restar a un negativo otro número negativo. Tampoco sabe realizar ninguna operación con más de de dos números ya sea suma, resta, multiplicación o división.

1.3 Segunda versión de la calculadora:

Si se compila los ficheros calcMejor.l y calcMejor.y y se ejecuta el archivo resultante se pueden observar los siguientes sucesos:

- Sabe realizar todas las operaciones que funcionaban correctamente en la primera versión. Sigue presentando los mismos problemas en las operaciones con enteros. No obstante, ahora si es capaz de realizar operaciones con más de dos números.

La razón por la que esto ocurre es que en el código de los ficheros calcMejor.l y calcMejor.y existe un diseño basado en la recursión de las operaciones por lo que estas operaciones están definidas en las estructuras del lenguaje y por tanto van bien, no como en el caso de la primera versión ya que no estaban declaradas, por consiguiente al ejecutarlas salía el mensaje syntax error.

Ejercicio 2

2.1 Descripción del ejercicio

El objetivo de este ejercicio es realizar una serie de mejoras sobre la calculadora de enteros positivos con fuentes calcMejor.l y calcMejor.y

1º mejora: Modificar la calculadora para que acepte enteros en decimal o en binario. Específicamente, una cadena que sólo tenga 0's y 1's y termine en la letra "b" (sin espacios en blanco en medio) debe ser interpretada como un entero en binario.

2º mejora: Modificar la calculadora para que todas las líneas de entrada terminen con ";" o bien ";b". Si la línea termina en ";b" el resultado se debe escribir en binario, si termina en ";"

3º mejora: Modificar la calculadora para que permita una única variable acumulador. Necesitamos permitir asignaciones a esta variable y referencias a la misma. La variable se llamará acum y se asigna de la forma "acum:= expresión ",

2.2 Código del ejercicio

1º Mejora

- Fichero ej21.y

```
/home/a737215/TeoComp/practica3/ej21.y - a737215@hendrix.cps.unizar.es - Editor - WinSCP
/* calcMejor.y fichero para la practica 3 de Teoria de la Computacion 2017/18 */
/*
 * Autor: Ruben Rodriguez
 * NIP: 737215
 * Práctica: 3
 * Ejercicio 2.1
 */
%{
#include <stdio.h>
%}
%token NUMBER EOL CP OP
%start calclist
%token ADD SUB
%token MUL DIV
%%

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

exp :
    factor
    | exp ADD factor { $$ = $1 + $3; }
    | exp SUB factor { $$ = $1 - $3; }
    | exp ADD NUMBER { $$ = $1 + $3; }
    | exp SUB NUMBER { $$ = $1 - $3; }
    | NUMBER { $$ = $1; }

factor :
    factor MUL factorsimple { $$ = $1 * $3; }
    | factor DIV factorsimple { $$ = $1 / $3; }
    | factorsimple MUL factorsimple { $$ = $1 * $3; }
    | factorsimple DIV factorsimple { $$ = $1 / $3; }

factorsimple : OP exp CP { $$ = $2; }
              | NUMBER
              ;

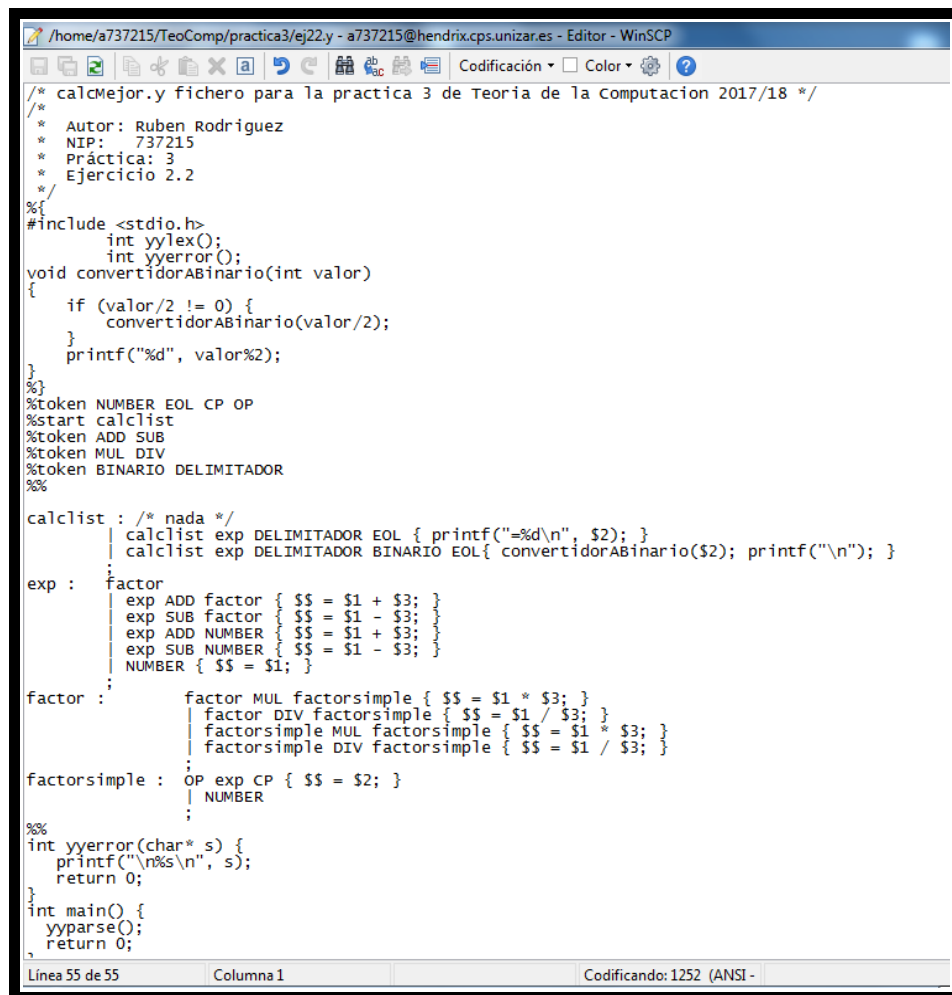
%%
int yyerror(char* s) {
    printf("\n%s\n", s);
    return 0;
}
main() {
    yyparse();
}
```

- Fichero ej21.l

```
/home/a737215/TeoComp/practica3/ej21.l - a737215@hendrix.cps.unizar.es - Editor - WinSCP
/* calcMejor.l fichero para la practica 3 de Teoria de la Computacion 2017/18 */
/*
 * Autor: Ruben Rodriguez
 * NIP: 737215
 * Práctica: 3
 * Ejercicio 2.1
 */
%{
#include "y.tab.h"
int cambio(int n)
{
    int decimalBinario = 0;
    int valor = 1;
    int digito;
    while (n!=0)
    {
        digito = n % 10;
        n = n / 10;
        decimalBinario = decimalBinario + (digito * valor);
        valor = 2 * valor;
    }
    return decimalBinario;
}
%}
%%
" + " {return(ADD);}
" - " {return(SUB);}
" * " {return(MUL);}
" / " {return(DIV);}
" ( " {return(OP);}
" ) " {return(CP);}
[0-9]+ {yylval = atoi(yytext); return(NUMBER);}
[0-9]+b {yylval = cambio(atoi(yytext)); return(NUMBER);}
\n {return(EOL);}
[ \t] { /* ignorar espacios */ }
. {return(yytext[0]); /* caracter inesperado */}
%%
```

2º Mejora

- Fichero ej22.y



```
/* calcmejor.y fichero para la practica 3 de Teoria de la Computacion 2017/18 */
/*
 * Autor: Ruben Rodriguez
 * NIP: 737215
 * Práctica: 3
 * Ejercicio 2.2
 */
%{
#include <stdio.h>
int yylex();
int yyerror();
void convertidorABinario(int valor)
{
    if (valor/2 != 0) {
        convertidorABinario(valor/2);
    }
    printf("%d", valor%2);
}
}%

%token NUMBER EOL CP OP
%start calclist
%token ADD SUB
%token MUL DIV
%token BINARIO DELIMITADOR
%%

calclist : /* nada */
        | calclist exp DELIMITADOR EOL { printf("%d\n", $2); }
        | calclist exp DELIMITADOR BINARIO EOL { convertidorABinario($2); printf("\n"); }

exp : factor
    | exp ADD factor { $$ = $1 + $3; }
    | exp SUB factor { $$ = $1 - $3; }
    | exp ADD NUMBER { $$ = $1 + $3; }
    | exp SUB NUMBER { $$ = $1 - $3; }
    | NUMBER { $$ = $1; }

factor :
        | factor MUL factorsimple { $$ = $1 * $3; }
        | factor DIV factorsimple { $$ = $1 / $3; }
        | factorsimple MUL factorsimple { $$ = $1 * $3; }
        | factorsimple DIV factorsimple { $$ = $1 / $3; }

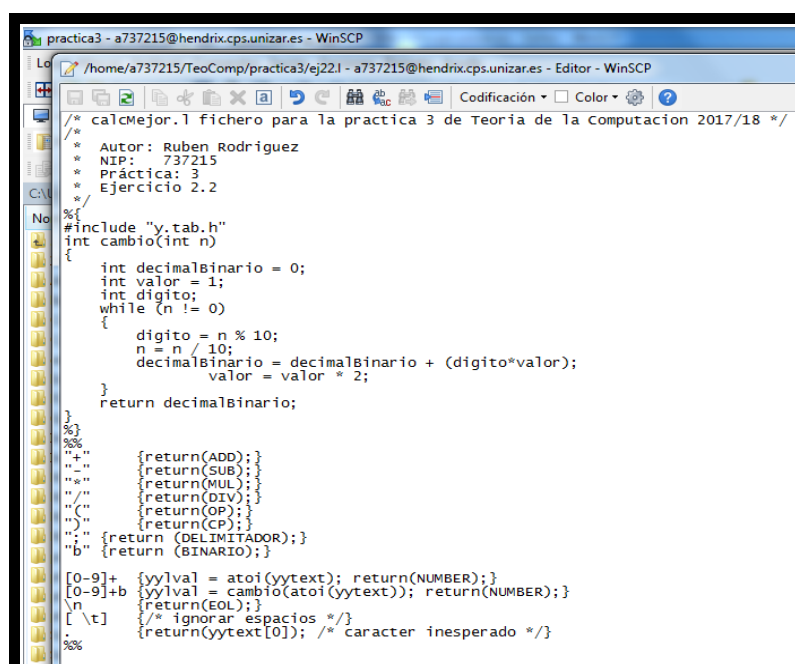
factorsimple : OP exp CP { $$ = $2; }
             | NUMBER
             ;

%%
int yyerror(char* s) {
    printf("\n%s\n", s);
    return 0;
}

int main() {
    yyparse();
    return 0;
}
```

Línea 55 de 55 Columna 1 Codificando: 1252 (ANSI -

- Fichero ej22.1



```
practica3 - a737215@hendrix.cps.unizar.es - WinSCP
/home/a737215/TeoComp/practica3/ej22.1 - a737215@hendrix.cps.unizar.es - Editor - WinSCP

/* calcmejor.1 fichero para la practica 3 de Teoria de la Computacion 2017/18 */
/*
 * Autor: Ruben Rodriguez
 * NIP: 737215
 * Práctica: 3
 * Ejercicio 2.2
 */
%{
#include "y.tab.h"
int cambio(int n)
{
    int decimalBinario = 0;
    int valor = 1;
    int digito;
    while (n != 0)
    {
        digito = n % 10;
        n = n / 10;
        decimalBinario = decimalBinario + (digito*valor);
        valor = valor * 2;
    }
    return decimalBinario;
}
}%

%%
"+" {return(ADD);}
"-" {return(SUB);}
"*" {return(MUL);}
"/" {return(DIV);}
"(" {return(OP);}
")" {return(CP);}
";" {return (DELIMITADOR);}
"b" {return (BINARIO);}

[0-9]+ {yyval = atoi(yytext); return(NUMBER);}
[0-9]+b {yyval = cambio(atoi(yytext)); return(NUMBER);}
\n {return(EOL);}
[ \t] {/* ignorar espacios */}
. {return(yytext[0]); /* caracter inesperado */}
%%
```

3º Mejora

- Fichero ej23.y

```
/home/a737215/TeoComp/practica3/ej23.y - a737215@hendrix.cps.unizar.es - Editor - WinSCP
/* calcMejor.y fichero para la practica 3 de Teoria de la Computacion 2017/18 */
/*
 * Autor: Ruben Rodriguez
 * NIP: 737215
 * Práctica: 3
 * Ejercicio 2.3
 */
%{
#include <stdio.h>
int total=0;
int yylex();
int yyerror();
}%
%token NUMBER EOL CP OP
%start calcList
%token ADD SUB
%token MUL DIV
%token ACUM ASIGN
%%
calcList : /* nada */
| calcList ACUM ASIGN exp EOL {total = $4;} // Asignar cualquier cosa
;
| calcList acum EOL {printf("%d\n", $2);}
;

acum : exp
| ACUM ADD exp {$$ = total + $3;}
| ACUM SUB exp {$$ = total - $3;}
| ACUM ADD NUMBER {$$ = total + $3;}
| ACUM SUB NUMBER {$$ = total - $3;}
//Multiplox
| ACUM MUL exp {$$ = total * $3;}
| ACUM DIV exp {$$ = total / $3;}

exp : factor
| exp ADD factor {$$ = $1 + $3;}
| exp SUB factor {$$ = $1 - $3;}
| exp ADD NUMBER {$$ = $1 + $3;}
| exp SUB NUMBER {$$ = $1 - $3;}
| NUMBER {$$ = $1;}

factor : factor MUL factorsimple {$$ = $1 * $3;}
| factor DIV factorsimple {$$ = $1 / $3;}
| factorsimple MUL factorsimple {$$ = $1 * $3;}
| factorsimple DIV factorsimple {$$ = $1 / $3;}
| factorsimple

factorsimple : OP exp CP {$$ = $2;}
| NUMBER
;

int yyerror(char* s) {
}
```

- Fichero ej23.l

```
/home/a737215/TeoComp/practica3/ej23.l - a737215@hendrix.cps.unizar.es - Editor - WinSCP
/* calcMejor.l fichero para la practica 3 de Teoria de la Computacion 2017/18 */
/*
 * Autor: Ruben Rodriguez
 * NIP: 737215
 * Práctica: 3
 * Ejercicio 3
 */
%{
#include "y.tab.h"
}%
%+
" + " {return(ADD);}
" - " {return(SUB);}
" * " {return(MUL);}
" / " {return(DIV);}
" ( " {return(OP);}
" ) " {return(CP);}
" acum " {return(ACUM);}
" : = " {return(ASIGN);}
[0-9]+ {yylval = atoi(yytext); return(NUMBER);}
\n {return(EOL);}
[ \t] /* ignorar espacios */
. {return(yytext[0]); /* caracter inesperado */}
%%
```

2.3.1 Conjunto de pruebas del ejercicio → primera mejora

```
bash-3.00$ bison -yd ej21.y
bash-3.00$ flex ej21.l
flex: can't open ej21.l
bash-3.00$ bison -yd ej21.y
bash-3.00$ flex ej21.l
bash-3.00$ gcc y.tab.c lex.yy.c -lfl -o ej21
```

```
bash-3.00$ ./ej21
10 + 3
=13
101b + 4
=9
7 - 5
=2
10b - 2
=0
```

2.3.2 Conjunto de pruebas del ejercicio → segunda mejora

```
bash-3.00$ bison -yd ej22.y
bash-3.00$ flex ej22.l
bash-3.00$ gcc y.tab.c lex.yy.c -lfl -o ej22
bash-3.00$ ./ej22
```

```
bash-3.00$ ./ej22
10 + 3;
=13
10 + 3;b
1101
1;b
1
8;b
1000
8 + 7;b
1111
```

2.3.3 Conjunto de pruebas del ejercicio → tercera mejora

```
bash-3.00$ bison -yd ej23.y
ej23.y: warning: 2 shift/reduce conflicts [-Wconflicts-sr]
ej23.y: warning: 21 reduce/reduce conflicts [-Wconflicts-rr]
bash-3.00$ flex ej23.l
bash-3.00$ gcc y.tab.c lex.yy.c -lfl -o ej23
bash-3.00$ ./ej23
cacion y division con la precedencia habitual.
```

```
acum:=4
acum
4
acum:=9
acum
9
acum:=2*3
acum
6
acum:=3*3
acum
9
acum *3
27
acum-15
-6
```

Ejercicio 3

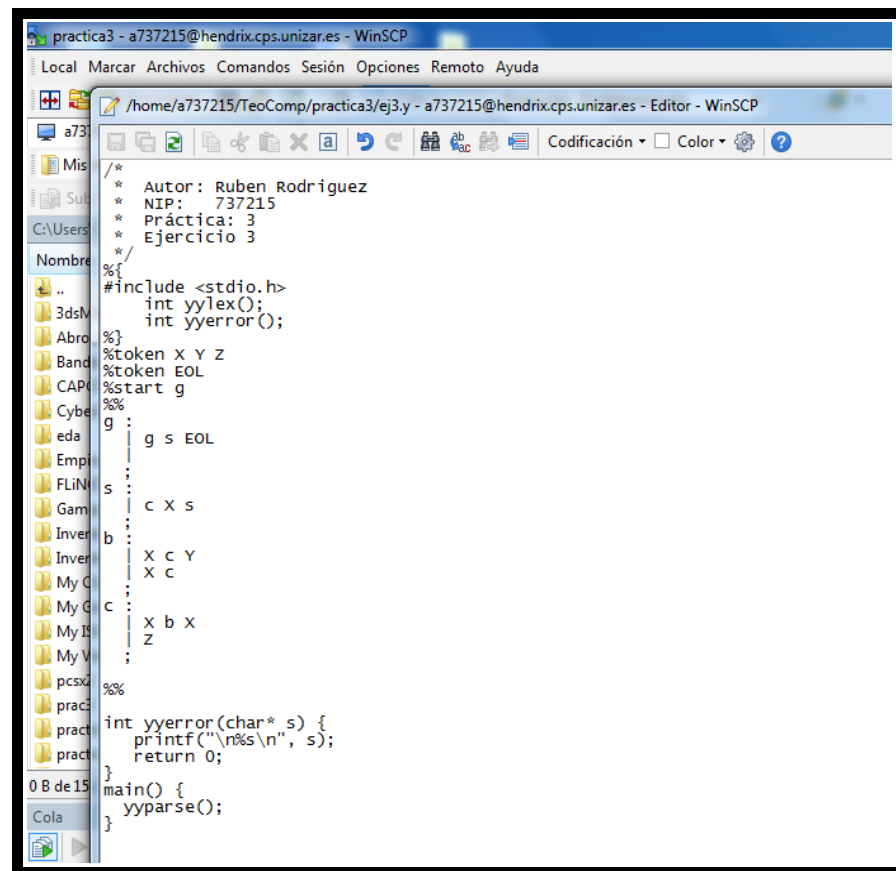
3.1 Descripción del ejercicio:

- Implementa un analizador sintáctico para la siguiente gramática:

$$S \rightarrow CxS \mid \text{epsilon}$$
$$B \rightarrow xCy \mid xC$$
$$C \rightarrow xBx \mid z$$

3.2 Código del ejercicio

- Fichero ej3.y

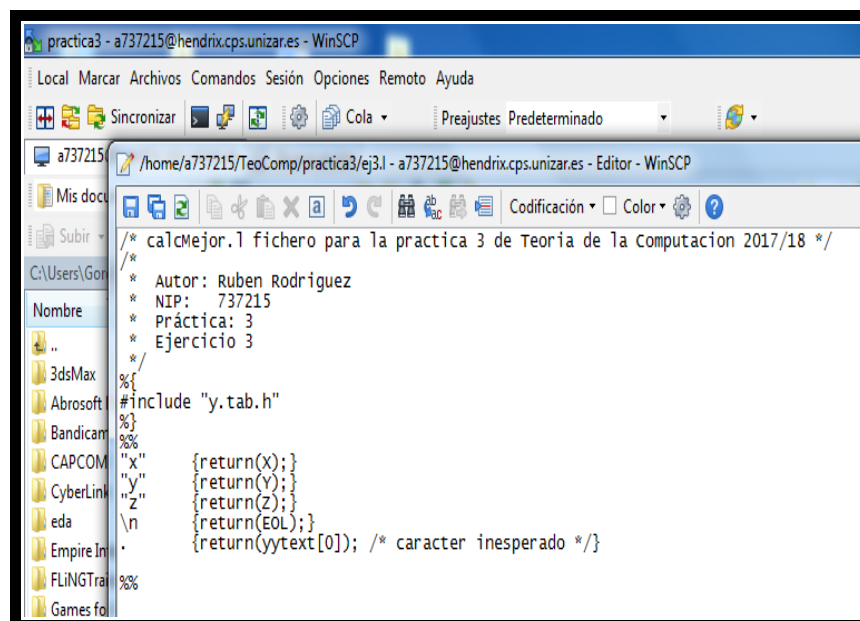
A screenshot of the WinSCP application window. The title bar reads 'practica3 - a737215@hendrix.cps.unizar.es - WinSCP'. The menu bar includes 'Local', 'Marcar', 'Archivos', 'Comandos', 'Sesión', 'Opciones', 'Remoto', and 'Ayuda'. The toolbar shows various file management icons. The left sidebar displays a file tree with folders like 'Mis documentos', 'Subir', and 'C:\Users\Gon'. The main editor pane shows the content of the file '/home/a737215/TeoComp/practica3/ej3.y'. The code is a C++ file with a header section containing author information and a main function that calls 'yyparse()'.

```
/*
 * Autor: Ruben Rodriguez
 * NIP: 737215
 * Práctica: 3
 * Ejercicio 3
 */
%{
#include <stdio.h>
int yylex();
int yyerror();
}%
%token X Y Z
%token EOL
%start g
%%
g :
  | g s EOL
  | s
  | c X s
  | b
  | X c Y
  | X c
  | c
  | X b X
  | Z
  ;
%%

int yyerror(char* s) {
    printf("\n%s\n", s);
    return 0;
}

main() {
    yyparse();
}
```

- Fichero ej3.l

A screenshot of the WinSCP application window. The title bar reads 'practica3 - a737215@hendrix.cps.unizar.es - WinSCP'. The menu bar includes 'Local', 'Marcar', 'Archivos', 'Comandos', 'Sesión', 'Opciones', 'Remoto', and 'Ayuda'. The toolbar shows various file management icons. The left sidebar displays a file tree with folders like 'Mis documentos', 'Subir', and 'C:\Users\Gon'. The main editor pane shows the content of the file '/home/a737215/TeoComp/practica3/ej3.l'. The code is a C++ file with a header section containing author information and a main function that calls 'yyparse()'.

```
/*
 * calCMejor.1 fichero para la practica 3 de Teoria de la Computacion 2017/18 */
/*
 * Autor: Ruben Rodriguez
 * NIP: 737215
 * Práctica: 3
 * Ejercicio 3
 */
%{
#include "y.tab.h"
}%
%%
"x" {return(X);}
"y" {return(Y);}
"z" {return(Z);}
\n {return(EOL);}
. {return(yytext[0]); /* caracter inesperado */}
%%
```

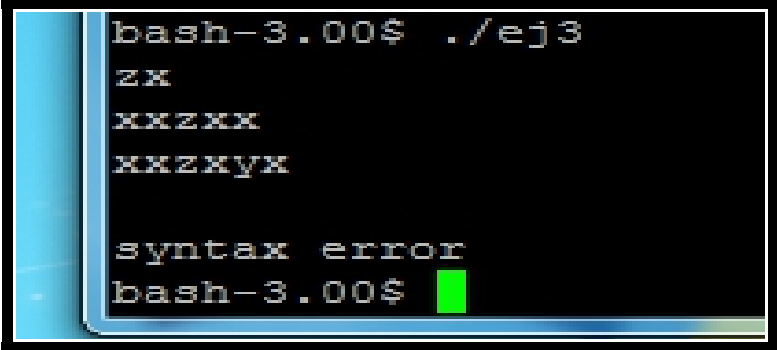
El lenguaje descrito con la gramática anterior es el siguiente:

- $L = ((xx)^m z (yx|x)^m x)^n$ tal que $m \geq 0$ y $n \geq 0$

Para ello, partimos de la primera regla $S \rightarrow CxS \mid \epsilon$. Si esta regla se repite varias veces queda una estructura igual a $(Cx)^n$. Posteriormente, C se puede convertir en $xxCy$, $xxCx$ o z . Dicha expresión la concatenamos con la anterior y se obtiene la expresión de arriba

3.3 Conjunto de pruebas del ejercicio

- Para llevar a cabo la compilación del programa solamente hay que escribir los comandos anteriores y después ejecutarlo



```
bash-3.00$ ./ej3
zx
xxzxx
xxzxyx

syntax error
bash-3.00$
```

Algunas palabras que pertenecen son zx repetida n veces, otra palabra es por ejemplo $xxzxx$, y en definitiva observando el lenguaje se pueden obtener todas ellas. Cuando aparece una palabra que no pertenece al lenguaje aparece el mensaje `syntax error`