

Análisis sintáctico de miniC

Curso 2024/2025

Enunciado de prácticas


- Página 12: tareas para implementar el analizador sintáctico
- Página 4: gramática libre de contexto de miniC
- Página 8: ejemplo de entrada

Tareas

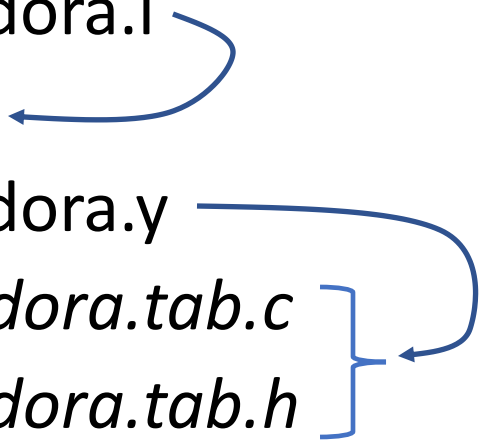
- Inclusión en el fichero *Flex* anterior de acciones para devolver **atributos** al analizador sintáctico.
- Crear el fichero *Bison*, que funcione conjuntamente con el analizador léxico, para reconocer sintácticamente ficheros descritos por la gramática de miniC. Para ello, será necesario:
 - Estudiar la necesidad de definir precedencias y asociatividades para los operadores aritméticos, con el fin de que las expresiones no sean ambiguas.
 - Añadir las acciones semánticas necesarias para generar la secuencia de reglas aplicadas para reconocer un programa de entrada.
- Estudiar los posibles conflictos desplazamiento/reducción y reducción/reducción.
- Realizar, opcionalmente, una recuperación de errores.
- **Verificar la corrección** del analizador sintáctico usando ficheros de prueba que deberán incluirse en la entrega de la práctica y comentarse en la memoria.

Ficheros para comenzar...

Analizador léxico de miniC


- lexico.l
 - *lex.yy.c*
 - lexico.h
 - main.c
 - makefile
 - entrada.txt
- 

Analizador de calculadora

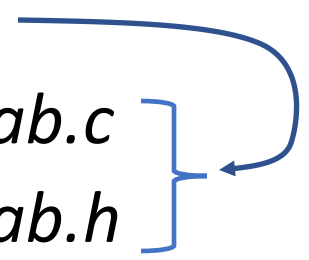
- calculadora.l
 - *lex.yy.c*
 - calculadora.y
 - *calculadora.tab.c*
 - *calculadora.tab.h*
 - main.c
 - makefile
 - entrada.txt
- 

¿Qué puedo aprovechar?

Analizador léxico de miniC

- lexico.l
 - lex.yy.c
 - ~~- lexico.h~~
 - ~~- main.c~~
 - ~~- makefile~~
 - entrada.txt
- 


Analizador de calculadora

- ~~- calculadora.l~~
 - ~~- lex.yy.c~~
 - calculadora.y
 - calculadora.tab.c
 - calculadora.tab.h
 - main.c
 - makefile
 - ~~- entrada.txt~~
- 

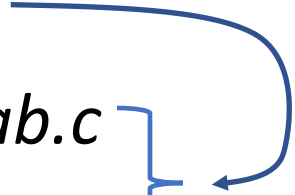
Aprovechar
el uso de
yyval para
pasar
lexemas al
analizador
sintáctico

Sugerencia de nombres...

Analizador léxico de miniC

- lexico.l
 - lex.yy.c
 - entrada.txt
- 

Analizador sintáctico de miniC

- sintactico.y
 - sintactico.tab.c
 - sintactico.tab.h
 - main.c
 - makefile
- 

Nombres de tokens

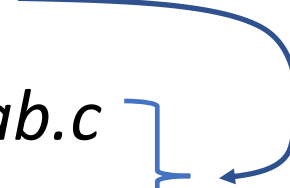
Analizador léxico de miniC

- lexico.l
- *lex.yy.c*
- entrada.txt

Usar los
nombres
de tokens
de lexico.l

Analizador sintáctico de miniC

- sintactico.y
- *sintactico.tab.c*
- *sintactico.tab.h*
- main.c
- makefile

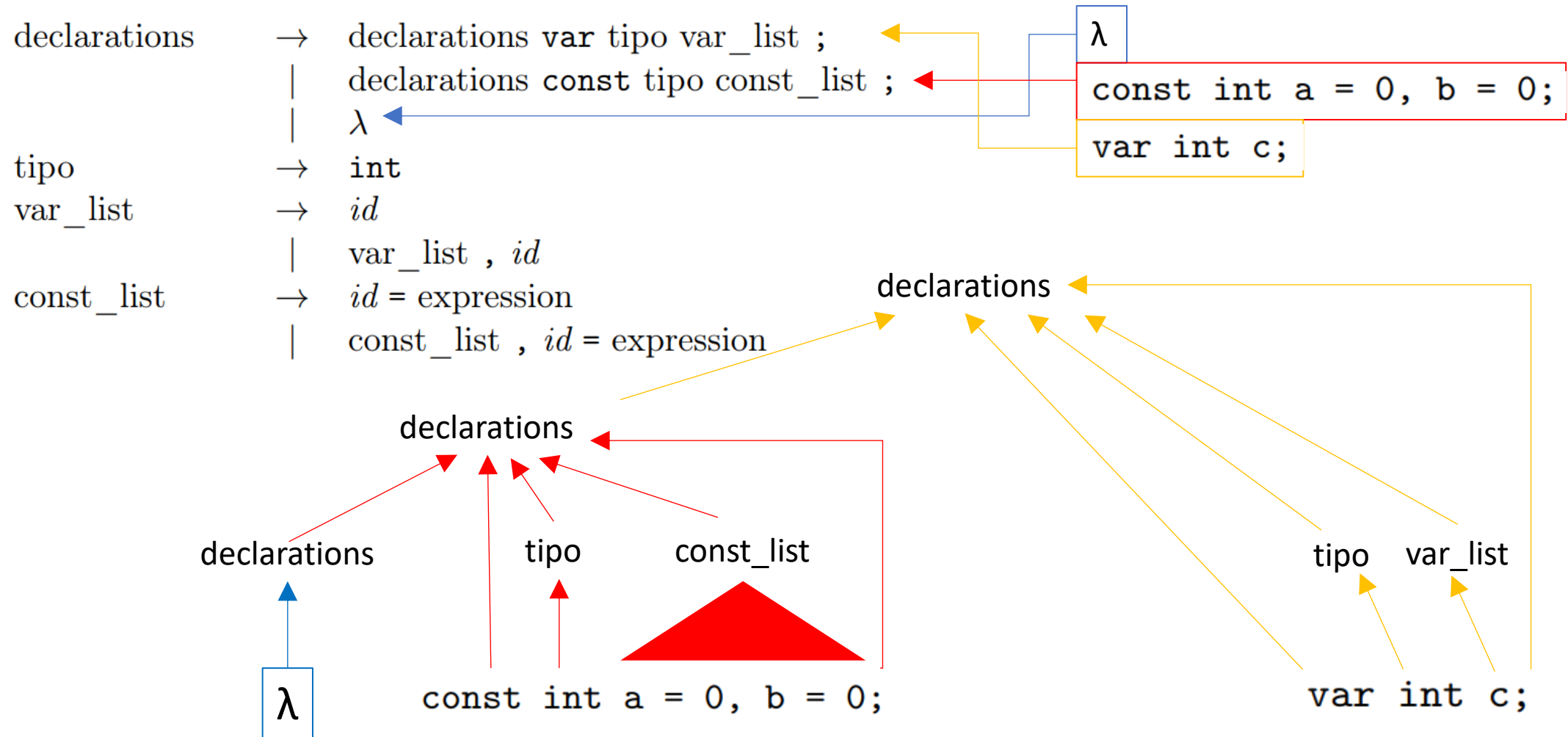


Sintaxis de miniC (1)

program \rightarrow *id* () { declarations statement_list }

```
prueba() {  
    const int a = 0, b = 0;  
    var int c;  
    print ("Inicio del programa\n");  
    c = 5+2-2;  
    if (a) print ("a","\n");  
    else if (b) print ("No a y b\n");  
        else while (c) {  
            print ("c = ",c,"\n");  
            c = c-2+1;  
        }  
    print ("Final","\n");  
}
```

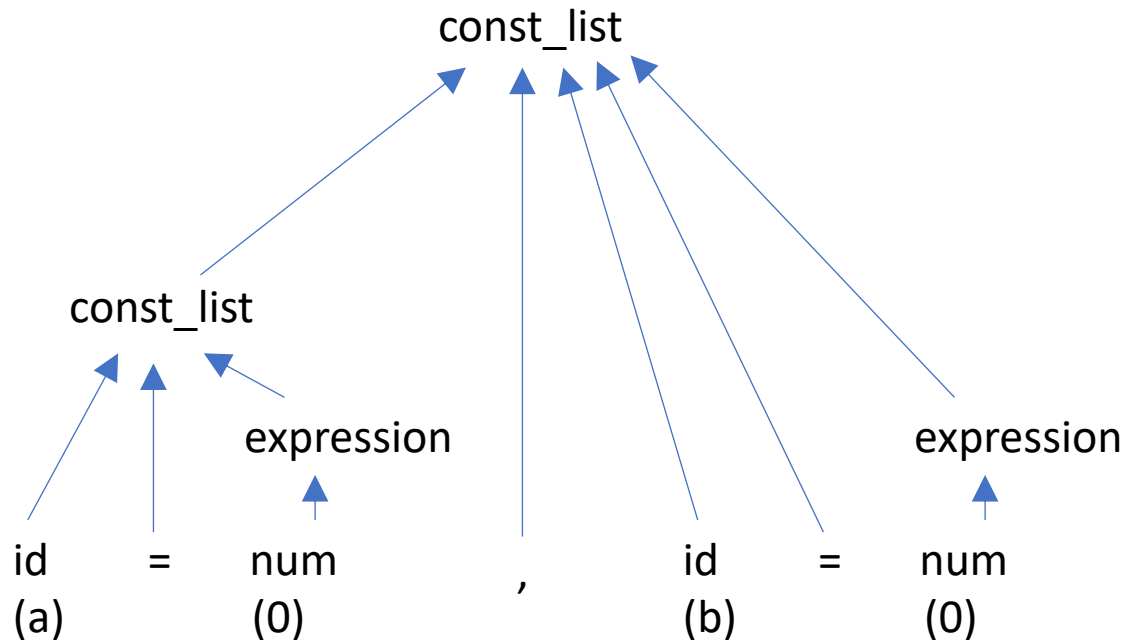

Sintaxis de miniC (2)



Sintaxis de miniC (3)

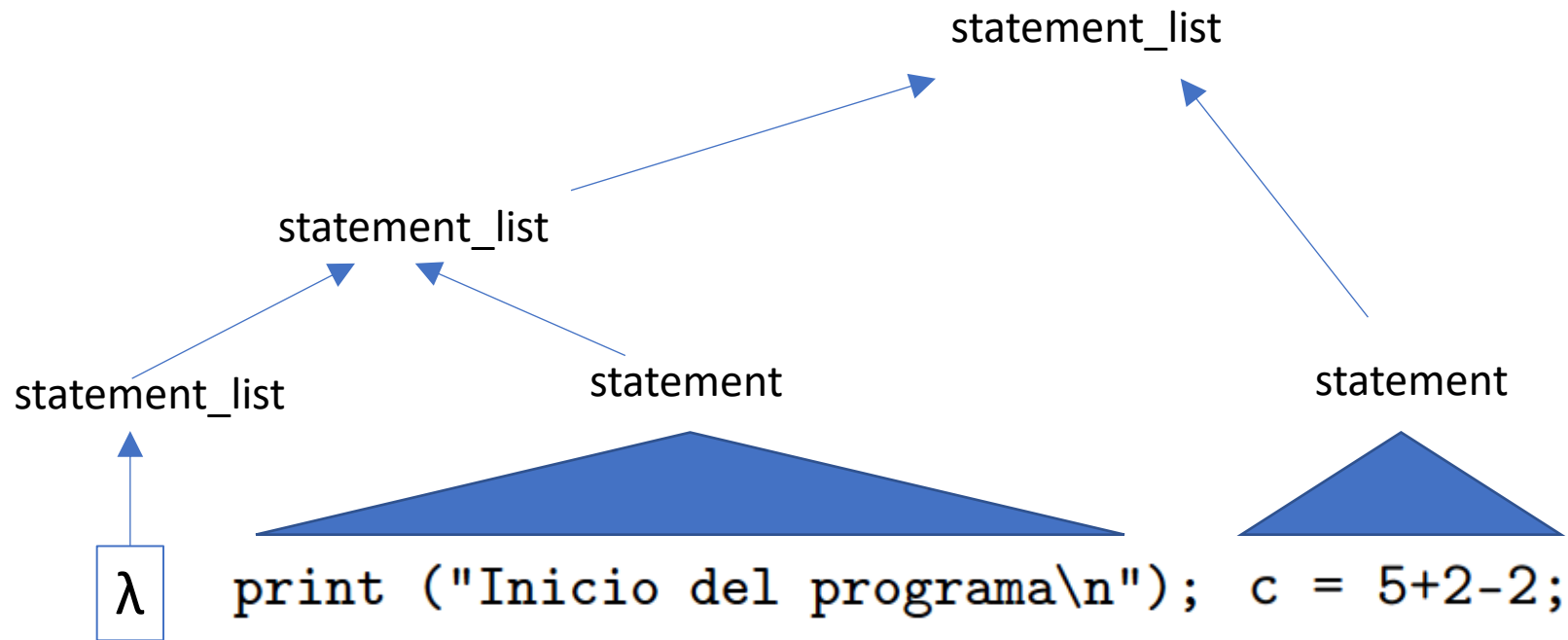
declarations → declarations var tipo var_list ;
| declarations const tipo const_list ;
| λ
tipo → int
var_list → id
| var_list , id
const_list → id = expression
| const_list , id = expression

const int a = 0, b = 0;



Sintaxis de miniC (4)

$$\begin{array}{lcl} \text{statement_list} & \rightarrow & \text{statement_list statement} \\ & | & \lambda \end{array}$$



Sintaxis de miniC (5)

statement → *id* = expression ;
 | { statement_list }
 | if (expression) statement else statement
 | if (expression) statement
 | while (expression) statement
 | print (print_list) ;
 | read (read_list) ;

The diagram illustrates nested code blocks with colored borders:

- A large black-bordered box contains the entire code snippet.
- A red-bordered box encloses the `if (b)` and `else while (c)` blocks.
- A green-bordered box encloses the `{` block of the `while (c)` loop.
- A purple-bordered box encloses the `print ("c = ", c, "\n");` statement.
- A yellow-bordered box encloses the `c = c-2+1;` statement.

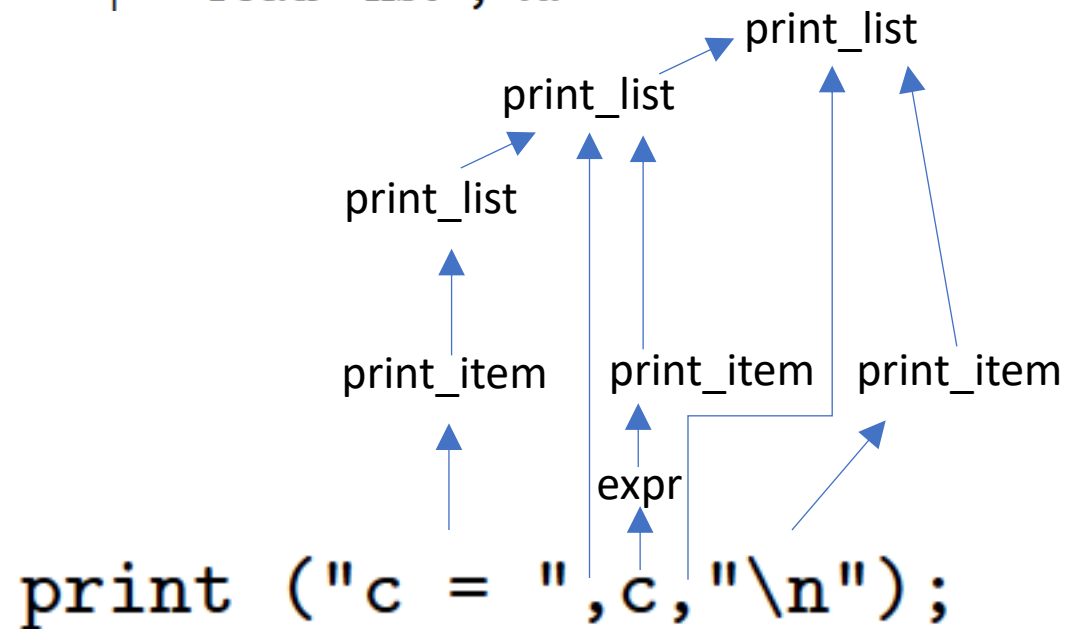
```
if (b) print ("No a y b\n");  
else while (c)  
{  
    print ("c = ", c, "\n");  
    c = c-2+1;  
}
```

Sintaxis de miniC (6)

`print_list` \rightarrow `print_item`
 | `print_list` , `print_item`

`print_item` \rightarrow `expression`
 | *string*

`read_list` \rightarrow *id*
 | `read list` , *id*



Sintaxis de miniC (7)

expression	→	expression + expression
		expression - expression
		expression * expression
		expression / expression
		(expression ? expression : expression)
		- expression
		(expression)
		<i>id</i>
		<i>num</i>