Instituto Tecnológico de Costa Rica

Departamento de Computación

Sede Cartago

Curso:

Compiladores e Intérpretes (IC-5701)

Profesor:

Esteban Arias Méndez

Periodo:

II

Fecha de entrega:

26-09-2016

Número de tarea:

4

Título de tarea:

Lex&Yacc + Micro

Estudiantes:

Carlos Adán Arguello Calderón-201173805

Luis Diego Flores Arguedas-201024948

Abstract:

Lex and Yacc are two useful tools in the process of creating parsers and scanners for recognition and construction of a language. The objective of this assignment is to use these tools and create an interpreter for the language MICRO.

Contents

[Definición y descripción de herramientas de Lex & Yacc 3](#_Toc462587466)

[1- Lex 3](#_Toc462587467)

[2- Yacc 5](#_Toc462587468)

[Bibliografía 6](#_Toc462587469)

# Definición y descripción de herramientas de Lex & Yacc

# Lex

Es un [programa](https://es.wikipedia.org/wiki/Programa_(computaci%C3%B3n)) para generar [analizadores léxicos](https://es.wikipedia.org/wiki/Analizador_l%C3%A9xico). Lex se utiliza comúnmente con el programa [Yacc](https://es.wikipedia.org/wiki/Yacc" \o "Yacc) que se utiliza para generar [análisis sintáctico](https://es.wikipedia.org/wiki/Analizador_sint%C3%A1ctico). Lex toma como entrada una especificación de analizador léxico y devuelve como salida el [código fuente](https://es.wikipedia.org/wiki/C%C3%B3digo_fuente) implementando el analizador léxico en [C](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_C).

La estructura de un archivo Lex se hace de esta manera:

Sección de declaraciones

%%

Sección de reglas

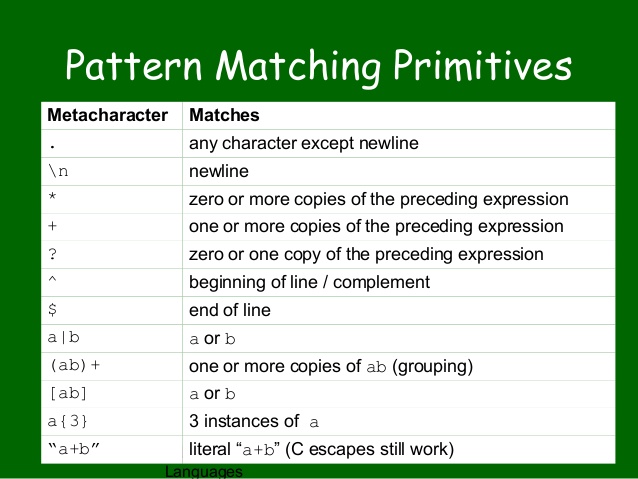
%%

Sección de código C

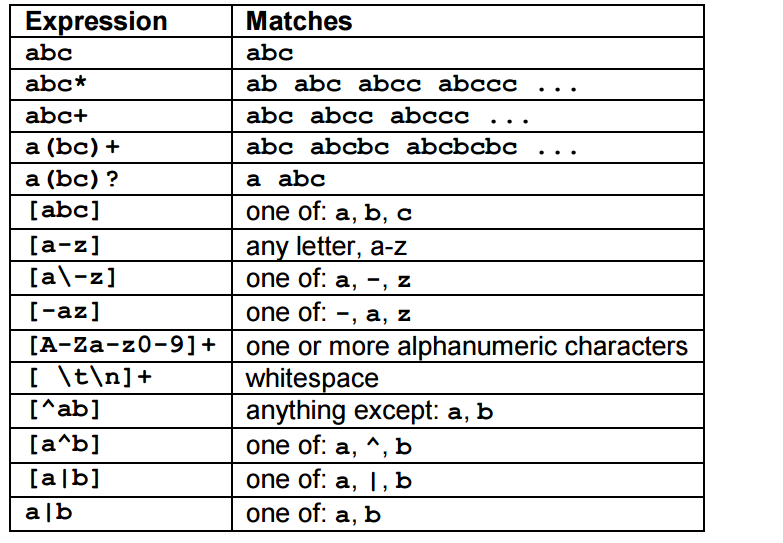
* La **sección de declaraciones** es el lugar para definir [macros](https://es.wikipedia.org/wiki/Macro) y para importar los archivos de cabecera escritos en [C](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_C). También es posible escribir cualquier código de C aquí, que será copiado en el archivo fuente generado. Este código en C debe ir entre los símbolos %{ %}.
* La **sección de reglas** es la sección más importante; asocia patrones a [sentencias](https://es.wikipedia.org/wiki/Sentencia) de C. Los patrones son simplemente [expresiones regulares](https://es.wikipedia.org/wiki/Expresi%C3%B3n_regular). Cuando el *lexer* encuentra un texto en la entrada que es asociable a un patrón dado, ejecuta el código asociado de C.
* La **sección de código C** contiene sentencias en C y [funciones](https://es.wikipedia.org/wiki/Funci%C3%B3n_(programaci%C3%B3n)) que serán copiadas en el archivo fuente generado. Estas sentencias contienen generalmente el código llamado por las reglas en la sección de las reglas. En programas grandes es más conveniente poner este código en un archivo separado y enlazarlo en tiempo de compilación.

Los siguientes cuadros ayudaran al trabajo de la creación del código Lex

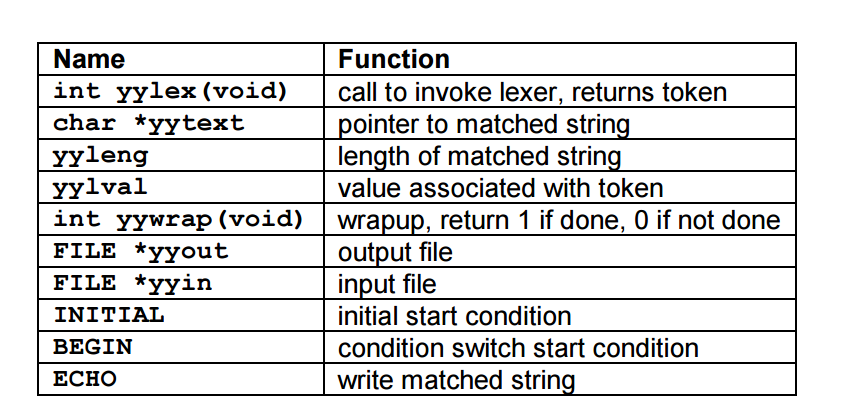
1. Primitivas de coincidencia de patrones



1. Ejemplos de coincidencias de patrones



1. Variables predefinidas de Flex

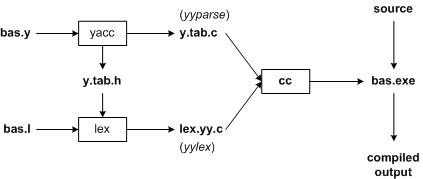


# Yacc

Programa informático de entrada general tiene algún tipo de estructura; de hecho, cada programa de computadora que hace de entrada puede ser pensado como la definición de un ''idioma de entrada'' aceptada por ésta. Un idioma de entrada puede ser tan complejo como un lenguaje de programación, o tan simple como una secuencia de números. Por desgracia, las instalaciones de entrada habituales son limitadas, difíciles de usar, y con frecuencia son laxos sobre la comprobación de sus entradas para la validez.

Yacc proporciona una herramienta general para la descripción de la entrada a un programa de ordenador. El usuario Yacc especifica las estructuras de su entrada, junto con el código que puede invocarse como se reconoce cada dicha estructura. Yacc convierte tal especificación en una subrutina que Han- DLE el proceso de entrada; Con frecuencia, es conveniente y apropiado tener la mayor parte del flujo de control en la aplicación del usuario a cargo de esta subrutina.

A continuación, esta imagen representa todo el proceso que hace Lex & Yacc para generar el scanner y parser:



Las gramáticas para Yacc son una variante de la notación Backus-Naur (BNF), Usada para expresar lenguajes libres de contexto. Mucho de los lenguajes modernos se pueden expresar en dicha notación. A continuación, se presenta un ejemplo de una expresión que multiplica y suma números:

E **->** E + E

E **->** E \* E

E **->** id

En el ejemplo anterior se denotan tres expresiones diferentes, los términos a la izquierda de la flecha como la expresión ‘E’ son denominado no terminales, ya que pueden producir los términos al lado derecho de la flecha. Lo términos como ‘id’ son expresiones terminales puesto que no generan ninguna otra expresión. La gramática mostrada anteriormente muestra las expresiones que pueden ser suma de dos expresiones, producto de dos expresiones o un identificador respectivamente. A continuación, se muestra como la gramática genera la expresión requerida.

E -> E \* E (r2)

-> E \* z (r3)

-> E + E \* z (r1)

-> E + y \* z (r3)

-> x + y \* z (r3)

# Bibliografía

1- http://dinosaur.compilertools.net