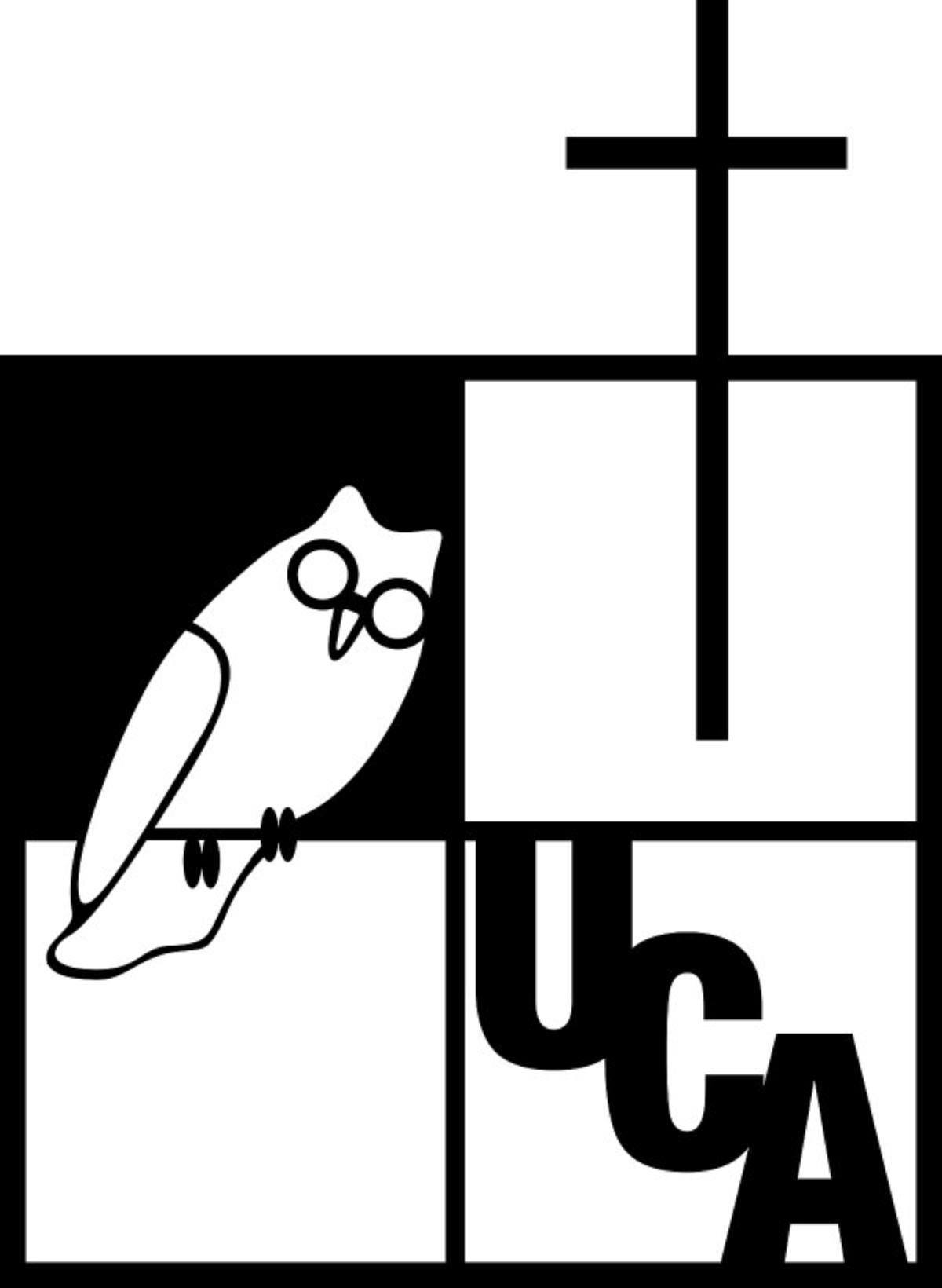
**Universidad Centroamericana**

**“José Simeón Cañas”**

**Facultad de Ingeniería y Arquitectura**

**Departamento de Electrónica e Informática**

****

**Teoría de Lenguajes de Programación**

**Proyecto final**

**Catedrático:**

Jaime Roberto Clímaco Navarrete

**Número del equipo:**

9

**Integrantes:**

José Alexander García Valladares 00062816

Raúl Antonio Granados Hernández 00138816

Marcelo Alexander Martínez Montenegro 00044616

Carlos Antonio Minero Dubón 00058016

**Fecha de entrega:**

Jueves 26 de noviembre del 2020

**Consideraciones:**

* Parser generado con Jison:
  + El código siempre debe tener una declaración de una función, es decir que, para que un código sea válido debería comenzar con una función *main*, por ejemplo.
  + El parser utiliza dos comillas consecutivas ("") para representar la palabra vacía.
  + Las palabras reservadas de while e if están limitadas a los siguientes operadores de relación: "<=", "<", ">" , ">=" , "!=".
  + Las palabras reservadas de while e if están limitados a los siguientes operadores lógicos: "&&","||", "~"
  + Es posible hacer uso de los siguientes operadores aritméticos: "+","-", "\*", "/","%".
  + Es posible hacer uso de paréntesis para agrupar las operaciones que se quieran realizar primero.
* Parser hecho de forma manual:
  + Solo se han considerado las operaciones aritméticas.

## BNF utilizadas para crear el parser con ayuda de Jison

program ::= comments declaration-list | comments macros declaration-list | macros declaration-list | declaration-list false

comments ::= COMENTARIO false

macros ::= MACRO IDENTIFICADOR CONSTANTE\_NUMERICA | MACRO IDENTIFICADOR CONSTANTE\_NUMERICA macros | MACRO IDENTIFICADOR CONSTANTE\_NUMERICA comments | INCLUDE < LIBRERIA > | INCLUDE < LIBRERIA > macros | INCLUDE < LIBRERIA > comments false

declaration-list ::= declaration-list declaration | declaration false

declaration ::= var-declaration | fun-declaration false

array-index ::= IDENTIFICADOR [ simple-expression ] | IDENTIFICADOR [ var ] false

var-declaration ::= type-specifier IDENTIFICADOR ; | type-specifier array-index ; | type-specifier IDENTIFICADOR = expression ; | type-specifier array-index = expression ; false

type-specifier ::= TIPO\_DATO | void false

fun-declaration ::= type-specifier IDENTIFICADOR ( params ) compound-stmt false

params ::= param-list | void false

param-list ::= param-list , param | param false

param ::= type-specifier IDENTIFICADOR | type-specifier IDENTIFICADOR [ ] false

compound-stmt ::= { local-declarations statement-list } false

local-declarations ::= local-declarations var-declaration | false

statement-list ::= statement-list statement | false

statement ::= comments | expression-stmt | compound-stmt | selection-stmt | iteration-stmt | return-stmt | declaration-list false

expression-stmt ::= expression ; | ; false

selection-stmt ::= if ( expression ) statement | if ( expression ) statement else statement false

iteration-stmt ::= while ( expression ) statement false

return-stmt ::= return ; | return expression ; false

expression ::= var = expression | simple-expression false

var ::= IDENTIFICADOR | IDENTIFICADOR [ expression ] false

simple-expression ::= additive-expression relop additive-expression | additive-expression logop simple-expression | additive-expression false

relop ::= <= | < | > | >= | != false

logop ::= && | || false

negop ::= ~ false

additive-expression ::= additive-expression addop term | negop term | term false

addop ::= + | - false

term ::= term mulop factor | factor false

mulop ::= \* | / | % false

factor ::= ( expression ) | var | call | CONSTANTE\_NUMERICA false

call ::= IDENTIFICADOR ( args ) false

args ::= arg-list | false

arg-list ::= arg-list , expression | expression false