**Universidad Tecnológica Centroamericana**

**Facultad de Ingeniería**

**Compiladores I**

*Documentación*

**Tic Tac Toe**

**Presentador por:**

Andrea José Mendoza Castañeda – 11711178

Daniel Vicenzo Trimarchi - 11741186

**Docente:**

Ing. Carlos Vallejo

**Fecha de Entrega:**

28 de septiembre de 2020

**Lugar:**

Tegucigalpa, M.D.C

**Tabla de Contenido**

[**Descripción de Lenguaje** 3](#_Toc52234596)

[**Distribución de Trabajo** 4](#_Toc52234597)

[**Uso de Lenguaje** 5](#_Toc52234598)

[**Ejemplos Correctos** 7](#_Toc52234599)

[*I. Ejemplo demostrando bloques de código anidados, declaración de variables y escritura.* 8](#_Toc52234600)

[*II. Ejemplo demostrando ciclos de repetición, declaración de arreglos de una y dos dimensiones y comentarios.* 9](#_Toc52234601)

[*III. Ejemplo demostrando declaración de main y otras funciones declaradas fuera de este y bloque de código switch.* 10](#_Toc52234602)

[**Ejemplos Incorrectos** 12](#_Toc52234603)

[*IV. Ejemplo I con Errores* 12](#_Toc52234604)

[*V. Ejemplo II con Errores* 13](#_Toc52234605)

[*VI. Ejemplo III con Errores* 14](#_Toc52234606)

[**Anexos** 16](#_Toc52234607)

[***I.*** ***Documento Flex Oficial del Lenguaje Tic Tac Toe*** 16](#_Toc52234608)

# **Descripción de Lenguaje**

El Lenguaje Tic Tac Toe nace como un intento de creación de un lenguaje inédito para la clase de Compiladores II. El nombre proviene del juego infantil popular “XO”, conocido como “Tic Tac Toe” en inglés. La mayoría de las palabras reservadas para abrir estructuras de nuestro lenguaje inician con el prefijo tic-. A su vez, las palabras reservadas para cerrar estructuras de nuestro lenguaje inician con el prefijo tac-. La palabra reservada “toe” se utiliza para cerrar funciones.

El archivo ejecutable del programa consiste en la unión de las clases generadas por Jflex y el Jcup, creando así un analizador léxico y sintáctico que reconoce los tokens y gramáticas de nuestro lenguaje. Así mismo, se incluye la creación de un árbol abstracto para identificar patrones de gramáticas y el manejo apropiado de errores—tanto semánticos como léxicos.

La experiencia de crear un nuevo lenguaje fue bastante interesante, pero a su vez tediosa y cansada. Varias de las herramientas utilizadas, si no todas, fueron completamente nuevas para nosotros. Se requirió de muchísima investigación para lograr presentar el archivo final.

# **Distribución de Trabajo**

En general se manejó una relación de trabajo simbiótica entre los dos miembros del equipo. Ambas partes se apoyaron mutuamente en el desarrollo de todas las tareas. Mientras uno programaba, el otro asesoraba y viceversa. A continuación, se presenta la división general de las tareas, pero cabe recalcar que, de igual manera, todos los miembros de los equipos involucrados estuvieron completamente involucrados en todas las etapas del desarrollo del proyecto.

|  |  |
| --- | --- |
| Tareas | Coordinador Encargado |
| Definición Inicial del Lenguaje | Andrea Mendoza |
| Archivo Flex y definición de Tokens | Andrea Mendoza |
| Gramáticas del Lenguaje e implementación a Cup | Daniel Trimarchi |
| Construcción de Árbol Sintáctico Abstracto | Daniel Trimarchi |
| Manejo de Errores | Daniel Trimarchi |
| Analizador Léxico | Andrea Mendoza |
| Documentación | Andrea Mendoza |

# **Uso de Lenguaje**

A continuación, se presenta la descripción de las estructuras principales de nuestro lenguaje seguidos de un ejemplo para la fácil comprensión.

**Para Asignar una Variable:** set [tipo] [id] as [valor]#

**Fin de instrucción:** #

**Variable de tipo Entero:** set int x as 9#

**Variable de tipo Boolean:** set bool flag as true#

**Arreglo de datos de una dimensión:** set int[] x#

set char[]x as ['a','b']#

**Arreglo de datos de dos dimensiones** set int[][] x#

set char[][]x as ['a','b'][‘c’,’d’]

**Variable de tipo Caracter:** set char opt as ‘a’#

**Comentario en línea:** !! esto es un comentario

**Comentario en párrafo:**

{! Esto

Es un comentario en

Párrafo !}

**Instrucción para leer input de usuario:** read

**Instrucción para escribir en pantalla:** write o writeln

**Escribir una variable:** write(x)# imprimiría el valor de x

**Escribir una cadena directa:** write (hola como estas)# imprimiría “hola como estas”

**Bloque de Decisión:**

ticif (condición)

elsif (condición)

tacif

**Bloque de Repetición:**

t4 (i as int#condicion#intervalo)

t4

ticwhile (cond)

tacwhile

**Bloque de decision tipo Switch:**

tichoose(var)

‘A’:  
 end#

‘B’:  
 end#

def:

end#

tachoose

**Funciones: (la palabra reservada toe significa FIN de función)**

func int f(…)#

func void main() =>

…

toe

func int f(…) =>  
…

toe

**Main:**

func void main() =>

set int x as 5#

toe

**TOKENS ADICIONALES:**

**Suma** +

**Resta** -

**Multiplicación** \*

**División** /

**Oprel** ( >, <, >=, <=, =, !=).

**And** &

**O**

# **Ejemplos Correctos**

## *I. Ejemplo demostrando bloques de código anidados, declaración de variables y escritura.*

func void main() =>

set int x as 5#

ticif (x > 5)

write (hola)#

else

write (adios)#

ticif (x < 2)

write(test)#

else

ticif (x==1)

write(x)#

tacif

tacif

tacif

toe

## *II. Ejemplo demostrando ciclos de repetición, declaración de arreglos de una y dos dimensiones y comentarios.*

func void main() =>

set char[]x as ['a','b']#

set int[][]y as [1,2,3,4,5]#

set int z as 3#

tic4 (int i as 0#i==10#i++)

ticwhile(z==3)

set char id#

read (id)# !! Esto es un comentario

tacwhile

{! Esto es

Otro comentario

De

Parrafo. !}

tac4

toe

## *III. Ejemplo demostrando declaración de main y otras funciones declaradas fuera de este y bloque de código switch.*

func void startA(bool flagA)#

func void startB(bool flagB)#

func void startC(bool flagC)#

func void main() =>

set char option#

read (option)#

tichoose(option)

'A':

set bool flagA as true#

end#

'B':

set bool flagB as true#

end#

'C':

set bool flagC as true#

end#

tachoose

toe

func void startA(bool flagA) =>

return 'A'#

toe

func void startB(bool flagB) =>

return 'B'#

toe

func void startC(bool flagC) =>

return 'C'#

toe

## **Ejemplos Incorrectos**

## *IV. Ejemplo I con Errores*

{! Ejemplo de error en declaración de variables, símbolo de lógica booleana dentro de un ticif y de falta de condición de un ticif !}

func void main() =>

set int x as #

set x as 10#

ticif (x=5)

write (hola)#

else

write (adios)#

ticif (x==2)

write(test)#

else

ticif

write(x)#

tacif

tacif

tacif

toe

Error mostrado:

Syntax Error at line 3, column 15 Unexpected token: DELIM |Syntax error

Syntax Error at line 4, column 6 Unexpected token: ID |Syntax error

Syntax Error at line 5, column 10 Unexpected token: IGUAL |Syntax error

Syntax Error at line 7, column 2 Unexpected token: ELSE |Syntax error

Syntax Error at line 11, column 3 Unexpected token: ELSE |Syntax error

--------------------------------------------

EXPECTED VARIABLE DECLARATION

EXPECTED TYPE IN VARIABLE DECLARATION

## *V. Ejemplo II con Errores*

{! Ejemplo de error que muestra falta de la palabra reservada “toe” al final de la función main !}

func void main() =>

set char[]x as ['a','b']#

set int[][]y as [1,2,3,4,5]#

set int z as 3#

tic4 (int i as 0#i==10#i++)

ticwhile(z==3)

set char id#

read (id)# !! Esto es un comentario

tacwhile

{! Esto es

Otro comentario

De

Parrafo. !}

tac4

Error mostrado:

Syntax Error : Part of declaration (main or function) is missing

## *VI. Ejemplo III con Errores*

{! Ejemplo de error de declaración de func al principio, error al cierre del código de bloque switch y falta de tipo de parámetros en las funciones fuera del main. !}

func void start(bool flagB)#

func void start(bool flagC)#

func void main() =>

set char option#

read (option)#

tichoose(option)

'A':

set bool flagA as true#

end#

'B':

set bool flagB as true#

end#

'C':

set bool flagC as true#

end#

toe

func void start(flagA) =>

return 'A'#

toe

func void start(flagB) =>

return 'B'#

toe

func void start(flagC) =>

return 'C'#

toe

Error mostrado:

Syntax Error at line 18, column 1 Unexpected token: CLOSEFUNC |Syntax error

Syntax Error at line 20, column 16 Unexpected token: PARIZQ |Syntax error

Syntax Error at line 24, column 16 Unexpected token: PARIZQ |Syntax error

Syntax Error at line 28, column 16 Unexpected token: PARIZQ |Syntax error

# **Anexos**

## ***Documento Flex Oficial del Lenguaje Tic Tac Toe***

package TicTacToe;

import java\_cup.runtime.\*;

import java\_cup.\*;

import java.io.IOException;;

import java\_cup.runtime.Symbol;

%%

%class TicTacCompiler

%unicode

%int

%cup

%char

%line

%column

%caseless

%{

int stCom = 0;

int ctLine;

private Symbol symbol(String namesym, int sym) {

return new Symbol(sym, yyline, yycolumn, namesym);

}

private Symbol symbol(String namesym, int sym, Object obj) {

return new Symbol(sym, yyline, yycolumn, obj);

}

private void error(String message) {

System.err.println("Error léxico en línea "+(yyline+1)+", columna "+(yycolumn+1)+" : "+message);

}

%}

letter = [a-zA-Z]

digit = [0-9]

space = (" "|"\t"|"\n")+

integer = {digit}+

sumres = "+"|"-"

multdiv = "\*"|"/"

boolex = "<"|">"|"<="|">="|"!="|"=="

boolVal = "true" | "false"

id = {letter}({letter}|{digit})\*

constchar = "'"{letter}"'"|"'"(" ")"'"

list = ({integer}",")\*{integer}

listc = ({constchar}",")\*{constchar}

array1D = ("int"|"char")"[]"

array2D = ("int"|"char")"[][]"

array1DVarI = "["{list}"]"

array1DVarC = "["{listc}"]"

arrayListI =({array1DVarI}",")\*{array1DVarI}

arrayListC =({array1DVarC}",")\*{array1DVarC}

array2DVarI = "["{arrayListI}"]"

array2DVarC = "["{arrayListC}"]"

null = "null"|"NULL"

%state COMMENT

%state LINECOMMENT

%state STRING

%%

<YYINITIAL>{

{array1D} { return symbol("arrayd", sym.ARRAYD, yytext()); }

{array2D} { return symbol("arraydd", sym.ARRAYDD,yytext()); }

{array1DVarI} { return symbol("arrayd", sym.ARRAYDVALI, yytext()); }

{array2DVarI} { return symbol("arraydd", sym.ARRAYDDVALI,yytext()); }

{array1DVarC} { return symbol("arrayd", sym.ARRAYDVALC, yytext()); }

{array2DVarC} { return symbol("arraydd", sym.ARRAYDDVALC,yytext()); }

"{!" { ctLine = yyline+1; stCom++; yybegin(COMMENT); }

"!}" { System.out.println("Missing {! for expression"); }

"!!" { yybegin(LINECOMMENT); }

"set" { return symbol("set", sym.SET); }

"as" { return symbol("as", sym.AS); }

"bool" { return symbol("bool", sym.BOOL); }

"void" { return symbol("void", sym.VOID); }

"func" { return symbol("func", sym.FUNC); }

"main" { return symbol("main", sym.MAIN); }

"write" { return symbol("write", sym.WRITE); }

"writeln" { return symbol("writeln", sym.WRITELN); }

"read" { return symbol("read", sym.READ); }

"#" { return symbol("#", sym.DELIM );}

"=>" { return symbol("=>", sym.OPENFUNC); }

"toe" { return symbol("toe", sym.CLOSEFUNC); }

"," { return symbol(",", sym.COMMA); }

"(" { return symbol("(", sym.PARIZQ); }

")" { return symbol(")", sym.PARDER, "hola"); }

"=" { return symbol("=", sym.IGUAL); }

"|" { return symbol("|", sym.OR); }

"&" { return symbol("&", sym.AND); }

"tic4" { return symbol("ticf4", sym.FOR); }

"tac4" { return symbol("ticf4", sym.CFOR); }

"ticwhile" { return symbol("tocwhile", sym.WHILE); }

"tacwhile" { return symbol("tacwhile", sym.CWHILE); }

"ticif" { return symbol("ticif", sym.IF); }

"elsif" { return symbol("elsif", sym.ELSIF); }

"else" { return symbol("else", sym.ELSE); }

"tacif" { return symbol("tacif", sym.CIF); }

"tichoose" { return symbol("tichoose", sym.SWITCH); }

"tachoose" { return symbol("tachoose", sym.CSWITCH); }

":" { return symbol(":", sym.CASE); }

"end" { return symbol("end", sym.ENDCASE); }

"def" { return symbol("def", sym.DEFCASE); }

"return" { return symbol("return", sym.RETURN); }

"break" { return symbol("break", sym.BREAK); }

"int" { return symbol("int", sym.INT); }

"char" { return symbol("char", sym.CHAR); }

{sumres} { return symbol("sumred", sym.SUMRES,yytext()); }

{multdiv} { return symbol("multdiv", sym.MULTDIV,yytext()); }

{boolex} { return symbol("boolex", sym.BOOLEX,yytext()); }

{null} { return symbol("null", sym.NULL, yytext()); }

{boolVal} { return symbol("boolVal", sym.BOOLVAL, yytext()); }

{constchar} { return symbol("char", sym.CHARVAL,yytext()); }

{integer} { return symbol("integer", sym.INTEGER, yytext()); }

{id} { return symbol("id", sym.ID, yytext()); }

{space} { }

. { error("Caracter inválido <"+ yytext()+"> en la línea:" + (yyline+1)); }

}

<COMMENT>{

"!}" { stCom--; if(stCom==0) yybegin(YYINITIAL); }

[^] { }

}

<LINECOMMENT>{

"\n" {yybegin(YYINITIAL);}

[^] { }

}