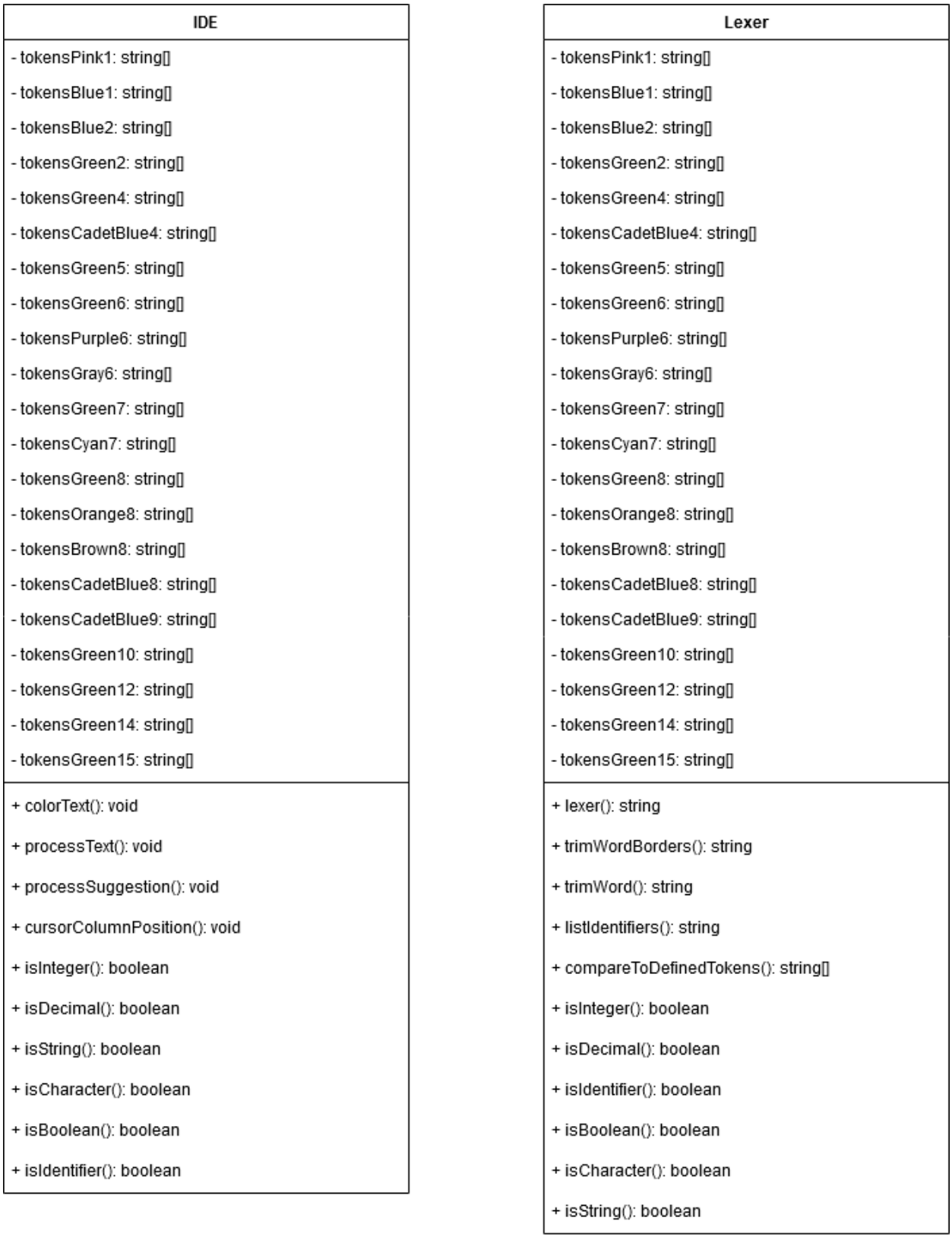
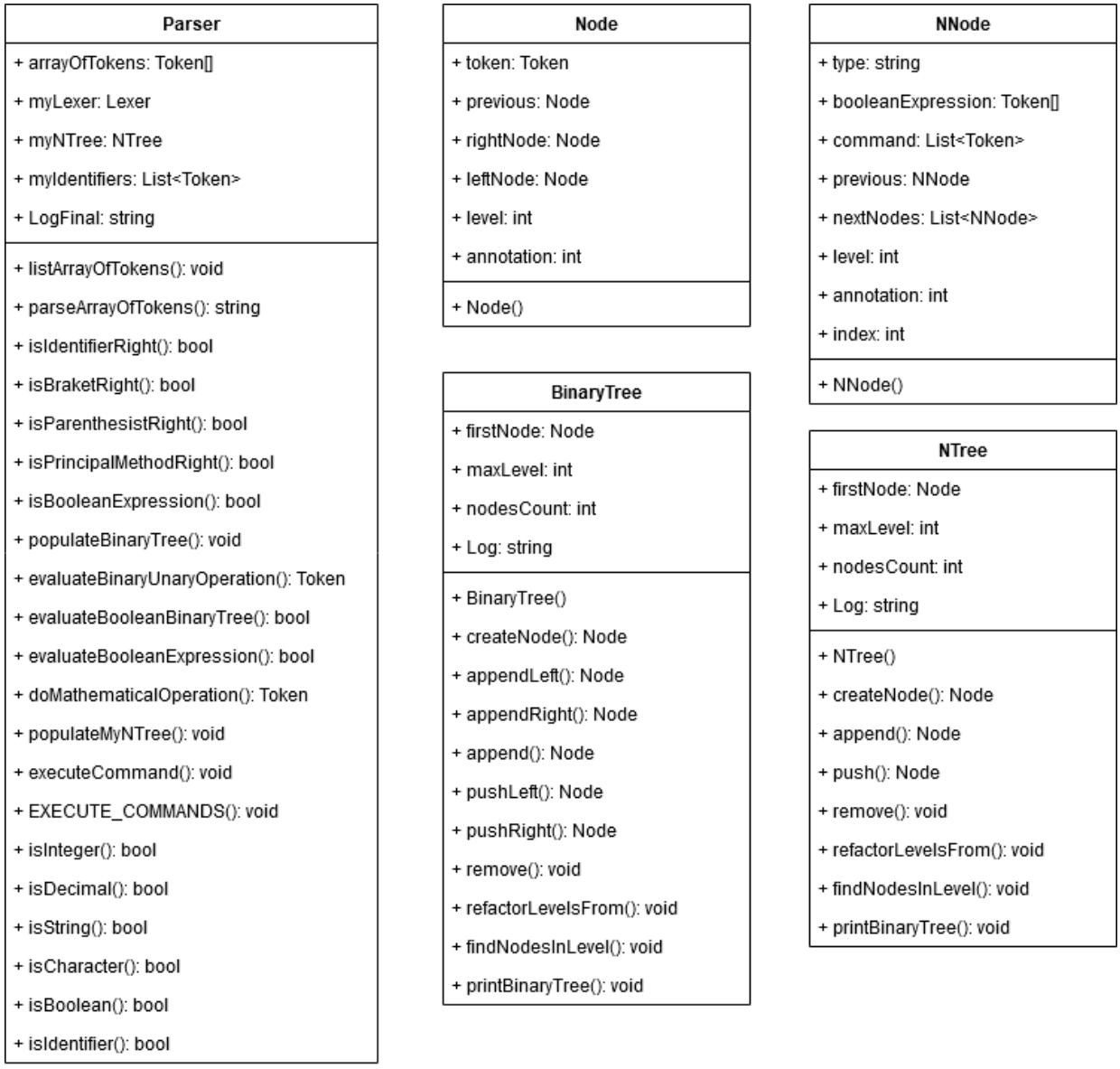
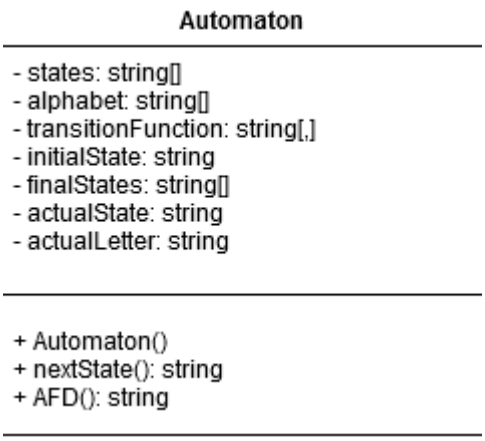
USAC/CUNOC
Ingeniería en Ciencias y Sistemas
Laboratorio de Lenguajes Formales y de Programación
Ronald Danilo Chávez Calderón
200130586

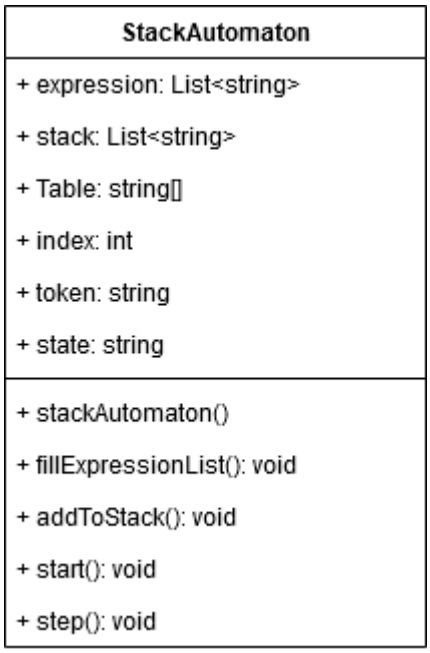
Manual Técnico

Diagrama de Clases




Parser
+ arrayOfTokens: Token[]
+ myLexer: Lexer
+ myNTree: NTree
+ myIdentifiers: List<Token>
+ LogFinal: string
+ listArrayOfTokens(): void
+ parseArrayOfTokens(): string
+ isIdentifierRight(): bool
+ isBraketRight(): bool
+ isParenthesistRight(): bool
+ isPrincipalMethodRight(): bool
+ isBooleanExpression(): bool
+ populateBinaryTree(): void
+ evaluateBinaryUnaryOperation(): Token
+ evaluateBooleanBinaryTree(): bool
+ evaluateBooleanExpression(): bool
+ doMathematicalOperation(): Token
+ populateMyNTree(): void
+ executeCommand(): void
+ EXECUTE_COMMANDS(): void
+ isInteger(): bool
+ isDecimal(): bool
+ isString(): bool
+ isCharacter(): bool
+ isBoolean(): bool
+ isIdentifier(): bool
Node
+ token: Token
+ previous: Node
+ rightNode: Node
+ leftNode: Node
+ level: int
+ annotation: int
+ Node()
BinaryTree
+ firstNode: Node
+ maxLevel: int
+ nodesCount: int
+ Log: string
+ BinaryTree()
+ createNode(): Node
+ appendLeft(): Node
+ appendRight(): Node
+ append(): Node
+ pushLeft(): Node
+ pushRight(): Node
+ remove(): void
+ refactorLevelsFrom(): void
+ findNodesInLevel(): void
+ printBinaryTree(): void
NNode
+ type: string
+ booleanExpression: Token[]
+ command: List<Token>
+ previous: NNode
+ nextNodes: List<NNode>
+ level: int
+ annotation: int
+ index: int
+ NNode()
NTree
+ firstNode: Node
+ maxLevel: int
+ nodesCount: int
+ Log: string
+ NTree()
+ createNode(): Node
+ append(): Node
+ push(): Node
+ remove(): void
+ refactorLevelsFrom(): void
+ findNodesInLevel(): void
+ printBinaryTree(): void


Automaton
- states: string[]
- alphabet: string[]
- transitionFunction: string[,]
- initialState: string
- finalStates: string[]
- actualState: string
- actualLetter: string
+ Automaton()
+ nextState(): string
+ AFD(): string


StackAutomaton
+ expression: List<string>
+ stack: List<string>
+ Table: string[]
+ index: int
+ token: string
+ state: string
+ stackAutomaton()
+ fillExpressionList(): void
+ addToStack(): void
+ start(): void
+ step(): void

VERSIÓN DE SOFTWARE

Se utilizó Visual Studio Community 2019 versión 16.7.7.
Windows Forms.

CLASE IDE

Atributos:

private tokensPink1: string[]
Tokens color rosado con 1 caracter de tamaño

private tokensBlue1: string[]
Tokens color azul con 1 caracter de tamaño

private tokensBlue2: string[]
Tokens color azul con 2 caracteres de tamaño

private tokensGreen2: string[]
Tokens color verde con 2 caracteres de tamaño

private tokensGreen4: string[]
Tokens color verde con 4 caracteres de tamaño

private tokensGreen5: string[]
Tokens color verde con 5 caracteres de tamaño

private tokensGreen6: string[]
Tokens color verde con 6 caracteres de tamaño

private tokensPurple6: string[]
Tokens color púrpura con 6 caracteres de tamaño

private tokensGray6: string[]
Tokens color gris con 6 caracteres de tamaño

private tokensGreen7: string[]
Tokens color verde con 7 caracteres de tamaño

private tokensCyan7: string[]
Tokens color cyan con 7 caracteres de tamaño

private tokensGreen8: string[]
Tokens color verde con 8 caracteres de tamaño

private tokensOrange8: string[]
Tokens color naranja con 8 caracteres de tamaño

private tokensBrown8: string[]
Tokens color cafe con 8 caracteres de tamaño

private tokensGreen10: string[]
Tokens color verde con 10 caracteres de tamaño

private tokensGreen12: string[]
Tokens color verde con 12 caracteres de tamaño

private tokensGreen14: string[]
Tokens color verde con 14 caracteres de tamaño

private tokensGreen15: string[]
Tokens color verde con 15 caracteres de tamaño

Métodos:

public IDE()
Método para crear el objeto IDE.

public colorText(): int
Método para dar color al texto del RichTextBox.

public compareToDefinedTokens(): string
Método para comparar un string de entrada con los tokens definidos previamente, devuelve el token que encuentre.

public compile(): string
Método para compilar el texto completo y devolver una lista de tokens hallados.

public processText(): void
Método para procesar cada línea de texto donde nos hallemos y colorearla adecuadamente.

public openFileGT(): void
Método para abrir los archivos de código fuente.

public openFileLogGTE(): void
Método para abrir los archivos de log en el IDE.

public openFileProjectGTP(): void
Método para abrir los archivos de proyecto.

public saveFileGT(): void
Método para guardar los archivos de código fuente

public saveFileLogGTE(): void
Método para guardar los archivos de log en el IDE.

public saveFileProjectGTP(): void
Método para guardar los archivos de proyecto.

public isString(): bool
Método para averiguar si el texto ingresado es un string.

public isCharacter(): bool
Método para averiguar si el texto ingresado es un booleano.

public isDecimal(): bool
Método para averiguar si el texto ingresado es un decimal.

public isInteger(): bool
Método para averiguar si el texto ingresado es un entero.

public isBoolean(): bool
Método para averiguar si el texto ingresado es un booleano.

CLASE LEXER

Atributos:

private tokensPink1: string[]
Tokens color rosado con 1 caracter de tamaño

private tokensBlue1: string[]
Tokens color azul con 1 caracter de tamaño

private tokensBlue2: string[]

Tokens color azul con 2 caracteres de tamaño

private tokensGreen2: string[]
Tokens color verde con 2 caracteres de tamaño

private tokensGreen4: string[]
Tokens color verde con 4 caracteres de tamaño

private tokensGreen5: string[]
Tokens color verde con 5 caracteres de tamaño

private tokensGreen6: string[]
Tokens color verde con 6 caracteres de tamaño

private tokensPurple6: string[]
Tokens color púrpura con 6 caracteres de tamaño

private tokensGray6: string[]
Tokens color gris con 6 caracteres de tamaño

private tokensGreen7: string[]
Tokens color verde con 7 caracteres de tamaño

private tokensCyan7: string[]
Tokens color cyan con 7 caracteres de tamaño

private tokensGreen8: string[]
Tokens color verde con 8 caracteres de tamaño

private tokensOrange8: string[]
Tokens color naranja con 8 caracteres de tamaño

private tokensBrown8: string[]
Tokens color cafe con 8 caracteres de tamaño

private tokensGreen10: string[]
Tokens color verde con 10 caracteres de tamaño

private tokensGreen12: string[]
Tokens color verde con 12 caracteres de tamaño

private tokensGreen14: string[]

Tokens color verde con 14 caracteres de tamaño

private tokensGreen15: string[]
Tokens color verde con 15 caracteres de tamaño

Métodos:

public Lexer()
Método para crear el objeto Lexer.

public Lexer(): string
Módulo de cálculo para obtener los tokens.

Private trimWordBorders(): string
Método para quitar los espacios de los bordes.

Private trimWord(): string
Método para eliminar todos los espacios de la palabra.

Private listIdentifiers(): string
Método para hacer una base de datos de identificadores.

Private compareToDefinedTokens(): string[]
Método para comparar los tokens con los tokens definidos.

public isString(): bool
Método para averiguar si el texto ingresado es un string.

public isCharacter(): bool
Método para averiguar si el texto ingresado es un booleano.

public isDecimal(): bool
Método para averiguar si el texto ingresado es un decimal.

public isInteger(): bool
Método para averiguar si el texto ingresado es un entero.

public isBoolean(): bool
Método para averiguar si el texto ingresado es un booleano.

CLASE PARSER

Atributos:

public arrayOfTokens: Token[]
Arreglo de todos los tokens analizados por el lexer.

Private myLexer: Lexer
Registro de los tokens analizados por el lexer.

Public myNTree: Ntree
Árbol N-ario base para los comandos.

Public myIdentifiers: List<Token>
Lista de identificadores y sus valores actuales.

Public LogFinal: String
Resultados que se muestran en pantalla con los comandos ejecutados.

Métodos:

public listArrayOfTokens(): void
Hace un arreglo de tokens y sus valores por defecto

Public parseArrayOfTokens(): string
Método principal para hacer un anális sintáctico del árbol generado.

Private isIdentifiersRight(): bool
Método para averiguar si los identificadores están correctos.

Private isBraketRight(): bool
Método para saber si los brakets están bien pareados.

Private isParenthesisRight(): bool
Método para saber si los paréntesis están bien pareados.

Private isPrincipalMethodRight(): bool
Método para saber si el método principal está correctamente definido.

Private isBooleanExpression(): bool
Método para saber si una expresión es booleana.

Public populateBinaryTree(): void
Método que llena un árbol binario de operaciones booleanas.

Public evaluateBinaryUnaryOperation(): Token
Método para evaluar operaciones binarias o unarias de tipo booleano.

Public evaluateBooleanBinaryTree(): bool
Método para conocer el resultado de un árbol booleano.

Public evaluateBooleanexpression(): bool
Método para evaluar una expresion booleana.

Private doMathematicalOperation(): Token
Método para hacer una operación matemática.

Public populateMyNTree(): void
Método para llenar un árbol n-ario.

Public executeCommand(): void
Método para ejecutar un comando.

Public EXECUTE_COMMANDS(): void
Método para ejecutar el árbol de comandos.

CLASE AUTOMATON

Atributos:

private states: string[]
Estados definidos del autómata.

private alphabet: string[]
Alfabeto de letras que se ingresan al autómata

private transitionFunction: string[,]
Función de transición que rige los cambios de estados del autómata.

private initialState: string

Estado inicial del autómata.

private actualState: string
Estado actual del autómata.

private actualLetter: string
Letra del alfabeto actual que se ingresa al autómata.

Métodos:

public Automaton()
Método para crear el objeto Automaton.

public nextState(): string
Método para pasar al siguiente estado, se ingresa un estado y se devuelve el siguiente.

public AFD(): string
Método para devolver el siguiente estado de un AFD.

CLASE STACK AUTOMATON

Atributos:

public expression: List<string>
Lista de tokens de la expresión definida.

public stack: List<string>
Lista de símbolos de la gramática que están en la pila.

public Table: string[,]
Tabla bidimensional de las definiciones de la sintaxis.

private index: int
Indice temporal.

public token: string
Token actual.

public state: string
Estado actual de la máquina de pila.

Métodos:

public StackAutomaton()
Método para crear el objeto StackAutomaton.

public fillExpressionList(): void
Método para llenar la Lista de Tokens de la expresión definida.

public addToStack(): void
Método para añadir los símbolos de la gramática LL1 a la pila.

public start(): void
Método para empezar.

public step(): void
Método para operar cada paso.