**Practica 1:**

**Desarrollo manual de un analizador léxico para un lenguaje sencillo**

**Youssef El Faqir El Rhazoui**

**Enrique Ávila Rodríguez**

1. **Clases léxicas del lenguaje**

Para el lenguaje descrito en la práctica hemos identificado las siguientes clases léxicas. En la siguiente tabla se describe brevemente cada clase.

|  |  |
| --- | --- |
| **Clase Léxica** | **Descripción** |
| **num** | Palabra reservada que indica números (enteros como flotantes) |
| **bool** | Palabra reservada para identificar valores booleanos (1, 0) |
| **punt\_coma** | Identifica el final de una sentencia |
| **ID** | Referencia el nombre de una variable o cte. |
| **sep\_sec** | Indica la separación entre la sección de declaración e instrucciones |
| **asignacion** | Referencia al operador de asignación |
| **op\_mas** | Referencia al operador de suma binario o al unario de signo |
| **op\_menos** | Referencia al operador de resta binario o al unario de signo |
| **op\_por** | Referencia al operador de multiplicación |
| **op\_div** | Referencia al operador de división |
| **op\_and** | Referencia al operador conjuntor |
| **op\_or** | Referencia al operador disyuntor |
| **op\_not** | Hace referencia al operador lógico para la negación |
| **op\_menor\_que** | Hace referencia al operador de comparación ‘<’ |
| **op\_mayor\_que** | Hace referencia al operador de comparación ‘>’ |
| **op\_menor\_igual** | Hace referencia al operador de comparación ‘<=’ |
| **op\_mayor\_igual** | Hace referencia al operador de comparación ‘>=’ |
| **op\_igual** | Hace referencia al operador de comparación ‘==’ |
| **op\_distinto** | Hace referencia al operador de comparación ‘!=’ |
| **lit\_num** | Describe cómo va a ser un número en nuestro lenguaje, tanto entero como decimal |
| **val\_true** | Palabra reservada para identificar el valor booleano True |
| **val\_false** | Palabra reservada para identificar el valor booleano False |
| **fin\_arch** | Identifica el final de fichero |
| **abre\_par** | Identifica la apertura de un paréntesis |
| **cierra\_par** | Identifica el cierre de un paréntesis |

1. **Especificación formal**

A continuación, pasamos a especificar las clases léxicas mediante expresiones regulares. Lo que son caracteres o cadenas lo hemos denotado entre comillas simples.

|  |  |  |
| --- | --- | --- |
| **Def. Aux** | **Clases Léxicas** | **Ignorables** |
| **letra** -> [a-z, A-Z] | **num** -> ‘num’ | **sep** -> white\_space | EOL | TAB |
| **dig** -> [0-9] | **bool** -> ‘bool’ |  |
| **p\_ent** -> (op\_mas | op\_menos | ε)(dig)+ | **punt\_coma**  -> ‘;’ |  |
| **p\_dec** -> ‘.’(dig)+ | **ID** -> letra(letra | dig | ‘\_’)\* |  |
| **p\_exp** -> (‘e’ | ‘E’)p\_ent | **sep\_sec** -> ‘&&’ |  |
|  | **asignación** -> ‘=’ |  |
|  | **op\_mas** -> ‘+’ |  |
|  | **op\_menos**  -> ‘-’ |  |
|  | **op\_por** -> ‘\*’ |  |
|  | **op\_div** -> ‘/’ |  |
|  | **op\_and** -> ‘and’ |  |
|  | **op\_or**  -> ‘or’ |  |
|  | **op\_not** -> ‘not’ |  |
|  | **op\_menor\_que** -> ‘<’ |  |
|  | **op\_mayor\_que**  -> ‘>’ |  |
|  | **op\_menor\_igual** -> ‘<=’ |  |
|  | **op\_mayor\_igual** -> ‘>=’ |  |
|  | **op\_igual** -> ‘==’ |  |
|  | **op\_distinto** -> ‘!=’ |  |
|  | **lit\_num** -> p\_ent (p\_dec | p\_exp | p\_dec p\_exp | ε) |  |
|  | **val\_true** -> ‘true’ |  |
|  | **val\_false** -> ‘false’ |  |
|  | **abre\_par** -> ‘(’ |  |
|  | **cierra\_par** -> ‘)’ |  |
|  | **fin\_arch**  -> EOF |  |

1. **Diagrama de transiciones**

Finalmente, en la ilustración 1 se muestra el autómata para la implementación. Las palabras reservadas se reconocen en una etapa posterior en la que se compara con un diccionario o un trie.

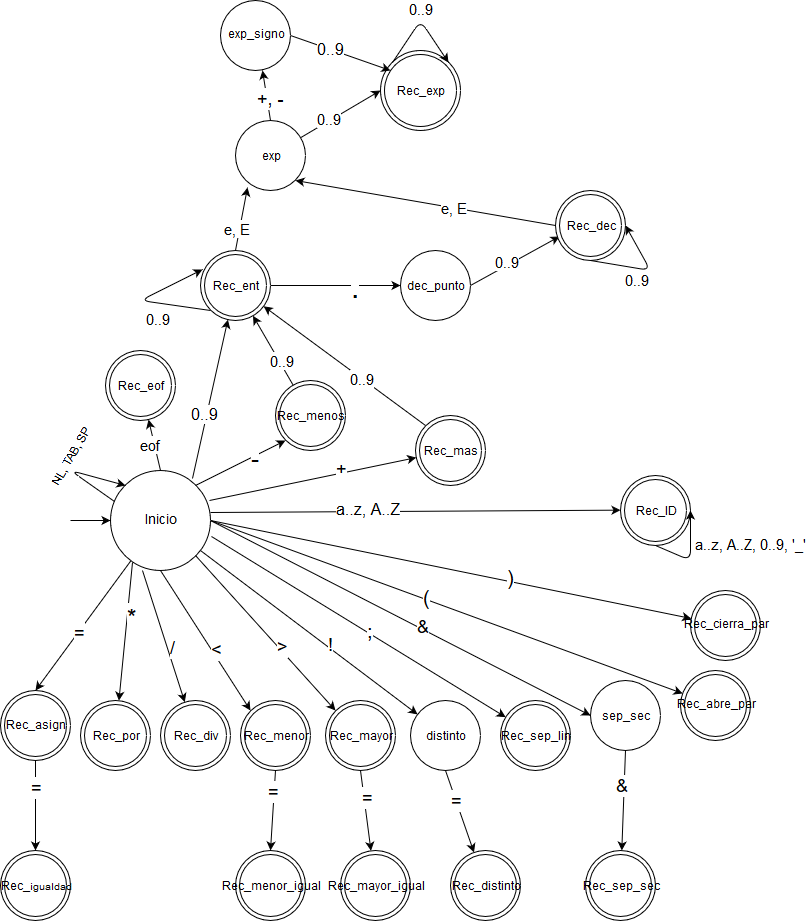


Ilustración 1: DFA