

INSTITUTO TECNOLÓGICO DE TIJUANA SUBDIRECCIÓN ACADÉMICA

DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

SEMESTRE AGOSTO-DICIEMBRE 2022

# CARRERA:

Ingeniería en Sistemas Computacionales

# MATERIA Y SERIE:

SCC-1015- Lenguaje y autómatas

# TÍTULO:

# Analizador Lexico

# UNIDAD:

# Unidades 3, 4, 5

# NOMBRE Y NÚMERO DE CONTROL:

Palacios Camacho Ruben Eduardo

# NOMBRE DEL MAESTRO:

Erasmo Estrada Peña

**FECHA DE ENTREGA:** 12 de mayo del 2023

# ¿Qué es un analizador Léxico?

El analizador léxico es la primera fase de un compilador, la función principal del analizador léxico es leer los caracteres de la entrada del programa fuente que se desea compilar, agruparlos en lexemas y producir como salida una secuencia de tokens para cada lexema encontrado en el programa fuente. El flujo de tokens es enviado al analizador sintáctico para su análisis. Con frecuencia el analizador léxico interactúa con la tabla de símbolos. Cuando el analizador léxico descubre un lexema que corresponde a un identificador, debe introducir dicho lexema en la tabla de símbolos; El analizador léxico se integra como una función de analizador sintáctico dentro de un compilador.

# Analizador Léxico:

|  |
| --- |
| %{  #include <stdio.h>  #include <ctype.h>  #include <string.h>  #include <stdlib.h>  #include <stdbool.h>  **struct** Nodo{*//Definicion de estructura de la tabla de simbolos*  **char** ID[31];  **struct** Nodo \*siguiente;  };  **struct** Tokens{*//Definicion de estructura de la tabla de tokens*  **char** Token[31];  **char** Atributo[31];  **struct** Tokens \*siguiente;  };  **struct** Nodo\*\* nodo = NULL; *//Apuntador de apuntadores*  **struct** Tokens\*\* token = NULL;  **int** VerificarID(**struct** Nodo\* nodo, **char** \*id){*//Verifica la posicion de un ID si esta dentro de la tabla de simbolos*  **int** pos = 0;      while (nodo != NULL){          if(strcmp(nodo->ID,id)== 0){              return pos;          }          pos++;          nodo = nodo->siguiente;      }      return -1;  }  **void** ImprimirTablaSimbolos(**struct** Nodo\* nodo){*//Imprimite tabla de simbolos*  **int** pos = 0;      printf("------------------------------- Tabla de Simbolos --------------------------------- \n\n");      printf("                               ID   |   Posicion                                    \n\n");      printf("----------------------------------------------------------------------------------- \n\n");      while (nodo != NULL){          printf("ID: %s   |   Posicion: %d\n",nodo->ID,pos );          pos++;          nodo = nodo->siguiente;      }  }  **void** ImprimirTablaTokens(**struct** Tokens\* nodo){*//Imprime la tabla de tokens*      printf("------------------------------- Tabla de Tokens ----------------------------------- \n\n");      printf("                          NombreToken   |   Atributo                                \n\n");      printf("----------------------------------------------------------------------------------- \n\n");      while (nodo != NULL){          printf("%s       |       %s\n",nodo->Token,nodo->Atributo);          printf("------------------------------------------------------------------ \n");          nodo = nodo->siguiente;      }  }  **void** agregarTokenLista(**struct** Tokens \*\*nodo, **char** \*Token,**char** \*atributo){*//Agrega Tokens a su tabla*  **struct** Tokens \*nuevonodo = malloc(sizeof(**struct** Tokens));      strcpy(nuevonodo->Token, Token);      strcpy(nuevonodo->Atributo,atributo);      if((\*nodo) == NULL){          \*nodo = nuevonodo;      }      else      {          agregarTokenLista(&(\*nodo)->siguiente,Token, atributo);      }  }  **bool** agregarIDLista(**struct** Nodo \*\*nodo, **char** \*id){ *//Agrega IDs a la tabla de simbolos*  **struct** Nodo \*nuevoNodo = malloc(sizeof(**struct** Nodo));      if(VerificarID(\*nodo,id) != -1){              return true;      }      else      {          strcpy(nuevoNodo->ID, id);          if((\*nodo) == NULL){              \*nodo = nuevoNodo;              return false;          }          else          {              agregarIDLista(&(\*nodo)->siguiente, id);          }      }  }  %}  %option noyywrap  %%  \+|\\*|\/|\-|\%|\\*\\*|\^|mul|div { agregarTokenLista(token,"Operador Aritmetico",yytext);      }  \=|\:|\+\=|\-\=|\\*\= { agregarTokenLista(token,"Asignacion",yytext);               }  \# { agregarTokenLista(token,"Comentarios","");                  }  \" { agregarTokenLista(token,"Comilla\_doble","");                }  \' { agregarTokenLista(token,"Comilla\_simple","");               }  \{ { agregarTokenLista(token,"Llave\_Izquierda","");              }  \} { agregarTokenLista(token,"Llave\_Derecha","");                }  \( { agregarTokenLista(token,"Parentecis\_Izq","");               }  \) { agregarTokenLista(token,"Parentecis\_Der","");               }  \] { agregarTokenLista(token,"Corchete\_Der","");                 }  \[ { agregarTokenLista(token,"Comilla\_Izq","");                  }  and|not|or|instanceof|sizeof|xor|add|sub|cmp|\=\=|\!\=|\<|\>|\||\<\=|\>\=|\<\>|equ|neq|gt|lt|geq|leq { agregarTokenLista(token,"Operadores",yytext);               }  [0-9]+ { agregarTokenLista(token,"Numero",yytext);                   }  private|protected|public|org|global|internal { agregarTokenLista(token,"Acceso\_Memoria",yytext);           }  \&\&|\?|\|\||\!|test { agregarTokenLista(token,"Operador\_Logico",yytext);          }  bagof|setof { agregarTokenLista(token,"Agrupacion",yytext);               }  false|true  { agregarTokenLista(token,"Booleano",yytext);                 }  for|while|do|loop { agregarTokenLista(token,"Ciclos",yytext);                   }  break|continue|endswitch { agregarTokenLista(token,"Controladores\_Ciclos",yytext);     }  extends|package|final|implement { agregarTokenLista(token,"Herencia",yytext);                 }  stream|ios|ifstream|ofstream|fstream { agregarTokenLista(token,"Modificadores\_Archivos",yytext);   }  elif|else|if|case { agregarTokenLista(token,"Condicionales",yytext);            }  get|getabstime|gethdrspace|getdatasamples|getdatasamplesize|getinterpmethod|getsamples|getqualitydes|getsampleusingtime|gettsafteratevent|gettsafterevent|gettsatevent|gettsbeforeatevent|gettsbeforeevent|gettsbetweenevents { agregarTokenLista(token,"Optencion\_Valores",yytext);        }  def|lambda|pass|return|yield|as|width|precision|fill|goto|clone|final|function|include|require|callable|include\_once|require\_once|trait|eval|fn|func|fun|jmp|jz|je|jne|pop|call|rat|clause|findall|notrace|trace|consult|once|repeat|char\_code|number\_chars|atom\_chars|sub\_atom|atom\_length|string\_concat|string\_chars|go|inc|jl|jg|jb|ja|defer { agregarTokenLista(token,"Funciones",yytext);                }  switch|struct|interface|endswitch { agregarTokenLista(token,"Estructura\_Datos",yytext);         }  showbase|showpoint|showpos|fixed|floatfield|writecomment|writedate|writekey|writekeyunit|writehistory|print|write|nl { agregarTokenLista(token,"Despliegue",yytext);               }  del|skipws|deletekey|deleterecord|unset|retract|retractall { agregarTokenLista(token,"Eliminacion",yytext);              }  cin|readcard|readkey|readkeycmplx|readkeydbl|readkeylonglong|readkeylongstr|readkeyunit|readrecord|read { agregarTokenLista(token,"Entrada\_Datos",yytext);            }  async|await { agregarTokenLista(token,"Escritura\_Concurrente",yytext);    }  assert|except|raise|try|catch|throw|throws|with|exception { agregarTokenLista(token,"Excepciones",yytext);              }  finally|enddeclare|endwhile { agregarTokenLista(token,"Finalizacion",yytext);             }  native|super|setf|flags|unset|fifo|lifo { agregarTokenLista(token,"Metodos",yytext);                  }  short|static|synchronized|staticfp|void|flush|register { agregarTokenLista(token,"Modificador",yytext);              }  abstract|from|import { agregarTokenLista(token,"Modulos",yytext);                  }  uppercase|touppercase { agregarTokenLista(token,"Caracteres\_Mayusculas",yytext);    }  streambuf|istream|ostream|iostream|ld|ldi|ldp|ldf|al|di|ote|otl|otu|rlo|ao { agregarTokenLista(token,"Operadores\_i\_o",yytext);           }  is|in|transient|this { agregarTokenLista(token,"Pertenencia",yytext);              }  left|right|adjustfield { agregarTokenLista(token,"Posicion",yytext);                 }  cout|cerr|dec|oct|hex|unitbuf|basefield|die|out|halt { agregarTokenLista(token,"Salida",yytext);                   }  char|enum|none|nonlocal|byte|int|float|long|int|string|decimal|volatile|auto|const|double|extern|signed|unsigned|typedef|empty|var|tuples|isset|val|db|dw|dd|is\_decimal|is\_byte|is\_float|is\_int|is\_long|is\_string { agregarTokenLista(token,"Tipo\_Dato",yytext);                }  stream|mov|movsb|movsw { agregarTokenLista(token,"Envio\_Datos",yytext);              }  compound|ground|atomic|unsigned { agregarTokenLista(token,"Verificacion",yytext);             }  stl|clc|stc|cmc|cld|std|cli|sti|rol|ror|shl|shr|jc|jnc { agregarTokenLista(token,"Ateracion\_Bit",yytext);            }  set { agregarTokenLista(token,"IniciarValor","");                 }  trace { agregarTokenLista(token,"Act\_Modo\_Dep","");                 }  push { agregarTokenLista(token,"insercion","");                    }  namespace { agregarTokenLista(token,"Definicion\_bloque","");            }  class|implement { agregarTokenLista(token,"Clase",yytext);                    }  new { agregarTokenLista(token,"Inicializacion","");               }  skips { agregarTokenLista(token,"Manejo\_flujo","");                 }  as { agregarTokenLista(token,"Alias\_Variables","");              }  atom\_chars { agregarTokenLista(token,"AtomoCaracteres","");              }  pass { agregarTokenLista(token,"Bloque\_Vacio","");                 }  string\_chars|string\_concat { agregarTokenLista(token,"Modificador\_Cadenas",yytext);      }  clone|setf { agregarTokenLista(token,"Clonacion",yytext);                }  \;|\,|\\_ { agregarTokenLista(token,"Separador",yytext);                }  pid { agregarTokenLista(token,"Control\_Proceso","");              }    [[:space:]]+  { */\* ignorar espacios en blanco \*/* }  [0-9a-zA-Z\_]+                                       {                                                          agregarIDLista(nodo,yytext);*//Agrega ID a tabla de simbolos*  **int** Busqueda = VerificarID(\*nodo, yytext); *//Busca la posicion*  **char** atributo[31]; *// Variable de atributo para la tabla de tokens*                                                          sprintf(atributo, "%d", Busqueda); *// Conversion de Int a Char delimitado a su tamaño*                                                          agregarTokenLista(token,"ID",atributo); *//Agrega Token a tabla de Tokens*                                                      }  .                                                   {}  %%  **int** main(**int** argc, **char** \*argv**[]**) {      nodo = malloc(sizeof(**struct** Nodo\*)); *// Definicion de memoria dinamica de la tabla de Simbolos*      \*nodo = NULL;      token = malloc(sizeof(**struct** Tokens\*)); *// Definicion de memoria dinamica de la tabla de Tokens*      \*token = NULL;      if (argc < 2) { *//Utilizacion de archivo de entrada*          printf("Uso: %s archivo\_de\_entrada\n", argv[0]);      }      FILE \*archivo = fopen(argv[1], "r");*//Inicializacion de archivo de entrada*      if (archivo == NULL) {*//Verificacion de apertura correcta de archivo*          printf("Error: no se pudo abrir el archivo de entrada.\n");          return 1;      }      yyin = archivo; *// Archivo a escanear*      yylex();      ImprimirTablaTokens(\*token);      ImprimirTablaSimbolos(\*nodo);      fclose(archivo);*//Cerradura de archivo*      return 0;  } |

# Mis expresiones regulares

|  |  |  |
| --- | --- | --- |
| **Token** | **Expresion Regular** | **Lexema** |
| Operador Aritmetico | \+|\\*|\/|\-|\%|\\*\\*|\^|mul|div | +,\*,/,-,%,\*\*,^,mul,div |
| Asignacion | \=|\:|\+\=|\-\=|\\*\= | =,:,+ =,- =,\* = |
| Comentarios | \# | # |
| Comilla\_doble | \" | " |
| Comilla\_simple | \' | ' |
| Llave\_Izquierda | \{ | { |
| Llave\_Derecha | \} | } |
| Parentecis\_Izq | \( | ( |
| Parentecis\_Der | \) | ) |
| Corchete\_Der | \] | ] |
| Comilla\_Izq | \[ | [ |
| Operadores | and|not|or|instanceof|sizeof|xor|add|sub|cmp|\=\=|\!\=|\<|\>|\||\<\=|\>\=|\<\>|equ|neq|gt|lt|geq|leq | and,not,or,instanceof,sizeof,xor,add,sub,cmp, = =,! =,<,>,|,< =,> =,<>,equ,neq,gt,lt,geq,leq |
| Acceso\_Memoria | private|protected|public|org|global|internal | private,protected,public,org,global,internal |
| Operador\_Logico | \&\&|\?|\|\||\!|test | &&,?,||,!,test |
| Agrupacion | bagof|setof | bagof,setof |
| Booleano | false|true | false,true |
| Ciclos | for|while|do|loop | for,while,do,loop |
| Controladores\_Ciclos | break|continue|endswitch | break,continue,endswitch |
| Herencia | extends|package|final|implements | extends,package,final,implements |
| Modificadores\_Archivos | stream|ios|ifstream|ofstream|fstream | stream,ios,ifstream,ofstream,fstream |
| Condicionales | elif|else|if|case | elif,else,if,case |
| Optencion\_Valores | get|getabstime|gethdrspace|getdatasamples|getdatasamplesize|getinterpmethod|getsamples|getqualitydes|getsampleusingtime|gettsafteratevent|gettsafterevent|gettsatevent|gettsbeforeatevent|gettsbeforeevent|gettsbetweenevents | get,getabstime,gethdrspace,getdatasamples,getdatasamplesize,getinterpmethod,getsamples,getqualitydes,getsampleusingtime,gettsafteratevent,gettsafterevent,gettsatevent,gettsbeforeatevent,gettsbeforeevent,gettsbetweenevents |
| Funciones | def|lambda|pass|return|yield|as|width|precision|fill|goto|clone|final|function|include|require|callable|include\_once|require\_once|trait|eval|fn|func|fun|jmp|jz|je|jne|pop|call|rat|clause|findall|notrace|trace|consult|once|repeat|char\_code|number\_chars|atom\_chars|sub\_atom|atom\_length|string\_concat|string\_chars|go|inc|jl|jg|jb|ja|defer | def,lambda,pass,return,yield,as,width,precision,fill,goto,clone,final,function,include,require,callable,include\_once,require\_once,trait,eval,fn,func,fun,jmp,jz,je,jne,pop,call,rat,clause,findall,notrace,trace,consult,once,repeat,char\_code,number\_chars,atom\_chars,sub\_atom,atom\_length,string\_concat,string\_chars,go,inc,jl,jg,jb,ja,defer |
| Estructura\_Datos | switch|struct|interface|endswitch | switch,struct,interface,endswitch |
| Despliegue | showbase|showpoint|showpos|fixed|floatfield|writecomment|writedate|writekey|writekeyunit|writehistory|print|write|nl | showbase,showpoint,showpos,fixed,floatfield,writecomment,writedate,writekey,writekeyunit,writehistory,print,write,nl |
| Eliminacion | del|skipws|deletekey|deleterecord|unset|retract|retractall | del,skipws,deletekey,deleterecord,unset,retract,retractall |
| Entrada\_Datos | cin|readcard|readkey|readkeycmplx|readkeydbl|readkeylonglong|readkeylongstr|readkeyunit|readrecord|read | cin,readcard,readkey,readkeycmplx,readkeydbl,readkeylonglong,readkeylongstr,readkeyunit,readrecord,read |
| Escritura\_Concurrente | async|await | async,await |
| Excepciones | assert|except|raise|try|catch|throw|throws|with|exception | assert,except,raise,try,catch,throw,throws,with,exception |
| Finalizacion | finally|enddeclare|endwhile | finally,enddeclare,endwhile |
| Metodos | native|super|setf|flags|unset|fifo|lifo | native,super,setf,flags,unset,fifo,lifo |
| Modificador | short|static|synchronized|staticfp|void|flush|register | short,static,synchronized,staticfp,void,flush,register |
| Modulos | abstract|from|import | abstract,from,import |
| Caracteres\_Mayusculas | uppercase|touppercase | uppercase,touppercase |
| Operadores\_i\_o | streambuf|istream|ostream|iostream|ld|ldi|ldp|ldf|al|di|ote|otl|otu|rlo|ao | streambuf,istream,ostream,iostream,ld,ldi,ldp,ldf,al,di,ote,otl,otu,rlo,ao |
| Pertenencia | is|in|transient|this | is,in,transient,this |
| Posicion | left|right|adjustfield | left,right,adjustfield |
| Salida | cout|cerr|dec|oct|hex|unitbuf|basefield|die|out|halt | cout,cerr,dec,oct,hex,unitbuf,basefield,die,out,halt |
| Tipo\_Dato | char|enum|none|nonlocal|byte|int|float|long|int|string|decimal|volatile|auto|const|double|extern|signed|unsigned|typedef|empty|var|tuples|isset|val|db|dw|dd|is\_decimal|is\_byte|is\_float|is\_int|is\_long|is\_string | char,enum,none,nonlocal,byte,int,float,long,int,string,decimal,volatile,auto,const,double,extern,signed,unsigned,typedef,empty,var,tuples,isset,val,db,dw,dd,is\_decimal,is\_byte,is\_float,is\_int,is\_long,is\_string |
| Envio\_Datos | stream|mov|movsb|movsw | stream,mov,movsb,movsw |
| Verificacion | compound|ground|atomic|unsigned | compound,ground,atomic,unsigned |
| Ateracion\_Bit | stl|clc|stc|cmc|cld|std|cli|sti|rol|ror|shl|shr|jc|jnc | stl,clc,stc,cmc,cld,std,cli,sti,rol,ror,shl,shr,jc,jnc |
| IniciarValor | set | set |
| Act\_Modo\_Dep | trace | trace |
| insercion | push | push |
| Definicion\_bloque | namespace | namespace |
| Clase | class|implements | class,implements |
| Inicializacion | new | new |
| Manejo\_flujo | skips | skips |
| Alias\_Variables | as | as |
| AtomoCaracteres | atom\_chars | atom\_chars |
| Bloque\_Vacio | pass | pass |
| Modificador\_Cadenas | string\_chars|string\_concat | string\_chars,string\_concat |
| Clonacion | clone|setf | clone,setf |
| Separador | \;|\,|\\_ | ;,",",\_ |
| Control\_Proceso | pid | pid |

# Texto de Prueba

import java.util.Scanner;

public class SaludoPersonalizado {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Ingrese su nombre: ");

String nombre = scanner.nextLine();

System.out.println("¡Hola, " + nombre + "! Bienvenido al programa.");

scanner.close();

}

}