Practica 1: Diseño del lenguaje

1.Introducción:

el objetivo de esta primera practica es la definición de un lenguaje de programación para el cual se va a diseñar un traductor, este lenguaje contendra las siguientes caracteristicas:

- tipos de datos minimos: entero, real, caracter, booleano y pila.
- Operaciones tipicas para cada tipo:
  - Entero, real: suma, resta, multiplicación y división.
  - Booleano: not, or, and y xor.
  - Pila: push, pop, top.
- Sentencia de asignación.
- Permitira expresiones aritmeticas logicas.
- Tendra sentencia de entrada y de salida.
- Dispondra de las siguientes estructuras de control:
  - IF_THEN_ELSE.
  - WHILE.
  - DO_UNTIL.
- La comprobación de tipos será fuertemente tipado.
- Las declaraciónes deben ir entre dos marcas de inicio y fin y deben declararse dentro de los bloques.
- El lenguaje adimitira las letras mayusculas y minisculas.
- La sintaxis del lenguaje va a ser similar al lenguaje C y las palabras reservadas seran en castellano.
- El lenguaje implementara procedimientos como subprogramas.

2.Descripción formal de la sintaxis del lenguaje:

A continuación se presenta la descripción del lenguaje en BNF:

<Programa> ::= <Cabecera_programa> <bloque>
<Cabecera_programa> ::= principal <ParIzq> <ParDer>

<ParIzq> := (

<ParDer> :=)

<bloque> ::= <inicio_de_bloque>
<Declar_de_variables_locales>
<Declar_de_subprogs>
<Sentencias>
<fin_de_bloque> |

<inicio_de_bloque>
<Declar_de_variables_locales>
<Sentencias>
<fin_de_bloque> |

<inicio_de_bloque>
<Declar_de_subprogs>
<Sentencias>
<fin_de_bloque> |

<inicio_de_bloque>
<Sentencias>
<fin_de_bloque>

<inicio_de_bloque> ::= {

<fin_de_bloque> ::= }

<Declar_de_variables_locales> ::= <Marca_ini_declar_variables>
<Variables_locales>
<Marca_fin_declar_variables>
<Declar_de_subprogs> ::= <Declar_de_subprogs> <Declar_subprog>
<Declar_subprog> ::= <Cabecera_subprograma> <bloque>

<Cabecera_subprog> ::= procedimiento <id><ParIzq><parametros><ParDer><bloque> |

procedimiento <id><ParIzq><ParDer><bloque>

<parametros>::= <parametros><coma><tipo><nombre> | <tipo><nombre>

<Variables_locales> ::= <Variables_locales> <Cuerpo_declar_variables>

| <Cuerpo_declar_variables>
<Cuerpo_declar_variables> ::= <tipo><lista_nombres><PYC>

<lista_nombres>::= <Lista_nombres><coma><id>

| <id>
<Marca_ini_declar_variables> := "@"

<Marca_fin_declar_variables> := "@"

<tipo>::=<tipo_simple> | <tipo_compuesto>

<tipo_simple>::= entero | real | caracter | booleano

<tipo_compuesto>::= pila <menor> <tipo_simple> <mayor>

<Sentencias> ::= <Sentencias> <Sentencia> | <Sentencia>
<Sentencia> ::= <bloque>
| <sentencia_asignacion>
| <sentencia_if>
| <sentencia_while>
| <sentencia_entrada>
| <sentencia_salida>
| <sentencia_return>
| <sentencia_hacer_hasta>
| <llamada_procedimiento>

<llamada_procedimiento>::=<id><ParIzq><expresiones><ParDer><PYC> |
<id><ParIzq><ParDer><PYC>

<sentencia_asignacion> ::= <id> = <expresion><PYC>
<sentencia_if> ::= si <ParIzq><expresion><ParDer> <Sentencia> |
si <ParIzq><expresion><ParDer> <Sentencia> <sentencia_else>
<sentencia_else> := sino <Sentencia>
<sentencia_while> ::= mientras <ParIzq><expresion><ParDer> <Sentencia>
<sentencia_entrada> ::= entrada <lista_nombres> <PYC>
<sentencia_salida> ::= salida <lista-exp-cad><PYC>
<sentencia_hacer_hasta> ::= hacer <Sentencia> hasta <expresion><PYC>
<expresiones> ::= <expresiones><coma> <expresion> | <expresion>
<expresion> ::= <ParIzq> <expresion> <ParDer>
| <op_unario> <expresion>
| <expresion> <op_binario> <expresion>
| <MasMenos> <expresion>
| <expresion> <MasMenos> <expresion>
| <expresion> <MasMas> <expresion>
| <id>
| <constante>
| <constante_pila>
<constante> ::= <numero> | <real> | '<letra>' | <boolean> | <constante_cadena>

<lista_constantes> ::= <lista_constantes><coma><constante> | <constante>

<constante_cadena>::= a |]a

<constante_pila>::= <corchete_izq> <lista_constantes> <corchete_der>

<real>::= <numero>.<numero>

<boolean>::= verdad | falso

<lista-exp-cad> ::= <lista-exp-cad><coma><exp-cad> | <exp-cad>

<exp-cad> ::= <expresion>| <constante_cadena>

<MasMenos> ::= + | -

<MasMas> ::= ++

<op_unario> ::= ! | # | & | --

<op_binario> ::= || | & | ^ | * | / | < | >= | > | <= | == | !=

<id> := <letra> <cadena> | <letra>

<PYC> := ;

<numero> ::= <numero><digito> | <digito>

<cadena> ::= <cadena><letra> | <cadena><digito> | <letra> | <digito> | _<cadena>

<letra> ::= a|b|..|z|A|B|..|Z

<digito> := 0|1|..|9

<corchete_izq> ::= [

<corchete_der> ::=]

3.Definición de la semántica en lenguaje natural:

como se ha menciando antes las palabras reservadas del lenguaje estan en castellano, el codigo siguiente muestra cómo se escriben todas las instrucciones posibles en el lenguaje:

- cabecera del programa:
  - principal()
- declaracion de variables:

```
@
        entero   ve ;
        real vf ;
        caracter  vc ;
        booleano  vl ;
        pila entero   pe, pe2 ;
        pila real pf, pf2 ;
        pila caracter  pc, pc2 ;
        pila booleano  pl ;
@
```

- leer datos:

```
        entrada  var1 [,var2,.....] ;
```

- condicion:

```
    si (var1 > valor)
    {   @
            int dato ;
        @
        dato= 2 ;
        dato= valor*20/dato ;
        }
        sino {
            var1= 1000.0 ;
        }
```

- mostrar datos:

```
        salida "valor =  ", var ;
```

- operaciones con pila:

```
        pf= pf++10.0 ;
        pc= pc++'#' ;
            si (#(pe++20) == 20)
                ve= #pe ;
```

```
sino
        pe= pe * pe2 ;
        pe= pe2 - pe ++ 10  * (20/2000) ;
```

4.Identificación de TOKENS:

Nombre Token	Expresión	Código	Atributo
PRINCIPAL	principal	256	0:principal
TIPO	entero | real | caracter | boolean	257	0:entero 1:real 2:caracter 3:boolean
PILA	pila	258	0:pila
CONST	\"[^"]+\" | [0-9]\.?[0-9]?* | '[a | ˥a]' | verdad | falso | pila	259	0:\"[^"]+\" 1:[0-9]\.?[0-9]?* 2:'[a | ˥a]' 4:verdad 5:falso
IDENT	[a-zA-Z][a-zA-Z0-9_]+	260	0: [a-zA-Z][a-zA-Z0-9_]+
ASIG	=	261	0 : =
MASMAS	++	262	0 : ++
MASMENOS	+ | -	263	0: + 1: -
OPBIN	* | / | || | && | ^ | == | != | < | <= | > | >=	264	0:* 1:/ 2:|| 3:&& 4:^ 5:== 6:!= 7:< 8:<= 9:> 10:>=
OPUNARIO	! | &| # | --	265	0: ! 1:& 2:# 3:--
PARIZQ	(	266	0: (
PARDER	)	267	0:)
INIBLQ	{	268	0: {
FINBLQ	}	269	0: }
DELIMITADOR	@	270	0: @
COMA	,	271	0: ,
PYC	;	272	0: ;
SI	si	273	0: si
SINO	sino	274	0: sino
MIENTRAS	mientras	275	0: mientras
HACER	hacer	276	0: hacer
HASTA	hasta	277	0: hasta
ENT	entrada	278	0: entrada
SAL	salida	279	0: salida
CORCHETEIZQ	[	280	0: [
CORCHETEDER	]	281	0:]
PROCEDIMIENTO	procedimiento	282	0: procedimiento

<Programa> ::= <Cabecera_programa> <bloque>
<Cabecera_programa> ::= PRINCIPAL PARIZQ PARDER
<bloque> ::= INIBLQ
<Declar_de_variables_locales>
<Declar_de_subprogs>
<Sentencias>
FINBLQ |

INIBLQ
<Declar_de_variables_locales>
<Sentencias>
FINBLQ |

INIBLQ
<Declar_de_subprogs>
<Sentencias>
FINBLQ |

INIBLQ
<Sentencias>
FINBLQ

<Declar_de_variables_locales> ::= DELIMITADOR
<Variables_locales>
DELIMITADOR
<Declar_de_subprogs> ::= <Declar_de_subprogs> <Declar_subprog>
<Declar_subprog> ::= <Cabecera_subprograma> <bloque>

<Cabecera_subprog> ::= PROCEDIMIENTO IDENT PARIZQ<parametros>PARDER<bloque> |

PROCEDIMIENTO IDENT PARIZQ PARDER<bloque>
<parametros>::= <parametros> COMA <tipo> IDENT | <tipo> IDENT

<Variables_locales> ::= <Variables_locales> <Cuerpo_declar_variables>

| <Cuerpo_declar_variables>
<Cuerpo_declar_variables> ::= TIPO <lista_nombres> PYC

<lista_nombres>::= <Lista_nombres> COMA IDENT

| IDENT

<tipo>:= TIPO | PILA
<Sentencias> ::= <Sentencias> <Sentencia> | <Sentencia>
<Sentencia> ::= <bloque>
| <sentencia_asignacion>
| <sentencia_if>
| <sentencia_while>
| <sentencia_entrada>
| <sentencia_salida>
| <sentencia_return>
| <sentencia_hacer_hasta>

| <llamada_procedimiento>

<llamada_procedimiento>::=IDENT PARIZQ <expresiones> PARDER PYC |

IDENT PARIZQ PARDER PYC

<sentencia_asignacion> ::= IDENT ASIG <expresion> PYC

<sentencia_if> ::= SI PARIZQ <expresion> PARDER <Sentencia> |

SI PARIZQ <expresion> PARDER <Sentencia> <sentencia_else>

<sentencia_else> := SINO <Sentencia>

<sentencia_while> ::= MIENTRAS PARIZQ <expresion> PARDER <Sentencia>

<sentencia_entrada> ::= ENTRADA <lista_nombres> PYC

<sentencia_salida> ::= SALIDA <lista-exp-cad> PYC

<sentencia_hacer_hasta> ::= HACER <Sentencia> HASTA <expresion> PYC

<expresiones> ::= <expresiones> COMA <expresion> | <expresion>

<expresion> ::= PARIZQ <expresion> PARDER
| OPUNARIO <expresion>
| <expresion> OPBIN <expresion>
| MASMENOS <expresion>
| <expresion> MASMENOS <expresion>
| <expresion> MASMAS <expresion>
| IDENT
| CONST
| <constante_pila>

<constante_pila> ::= CORCHETEIZQ <lista_constantes> CORCHETEDER

<lista_constantes> ::= <lista_constantes> COMA CONST | CONST

<lista-exp-cad> ::= <lista-exp-cad> COMA <exp-cad> | <exp-cad>

<exp-cad> ::= <expresion>| <constante_cadena>

<cadena> ::= <cadena><letra> | <cadena><digito> | <letra> | <digito> | _<cadena>

<letra> ::= a|..|z|A|..|Z

<digito> := 0|..|9