**UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS**

(Universidad del Perú, DECANA DE AMÉRICA)

**FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA**



**ESTRUCTURA DEL LENGUAJE DE PROGRAMACIÓN**

**EAP : Ingeniería de Sistemas**

**Curso : Lenguajes y Compiladores**

**Docente : Ruiz De La Cruz Melo, Carlos Augusto**

**Integrantes :**

**Calderon Claudio, Eber Andres 18200214**

**Avellaneda Pinto, Diego 15200006**

**Coronel Ramirez, Patrick Miguel 18200143**

**Semestre : 2020 - ll**

**LIMA - PERÚ**

**ÍNDICE**

[**INTRODUCCIÓN**](#_heading=h.3rs8n4jqfze7) **2**

[**CAPÍTULO I: DESCRIPCIÓN DEL LENGUAJE DE PROGRAMACIÓN**](#_heading=h.109t5nodjag) **3**

[**ESTRUCTURA DE UN PROGRAMA**](#_heading=h.30j0zll) **3**

[**IDENTIFICADORES**](#_heading=h.pymrlpca9nsq) **5**

[Constantes](#_heading=h.udszup1zviaj) 5

[Variables](#_heading=h.bucvyq9p7dqe) 6

[Vectores de una dimensión](#_heading=h.ns536lu8huda) 7

[Estructurados (Clases, atributos y métodos)](#_heading=h.fyt88s5nedt4) 8

[**OPERADORES**](#_heading=h.1wlu6amjfdfn) **9**

[**EXPRESIONES**](#_heading=h.ictytokwg3sk) **11**

[**INSTRUCCIONES**](#_heading=h.u3t2savtil1g) **11**

[Asignación](#_heading=h.9gpz0fka80tz) 11

[Lectura / Escritura por pantalla](#_heading=h.sxo9wfbe6yd) 12

[**SENTENCIAS DE CONTROL DE FLUJO**](#_heading=h.fotcwl3faeu3) **13**

[6.1. Man - mana](#_heading=h.a4l6yqdxouj) 13

[6.2. Kama](#_heading=h.xomulagkiogn) 14

[6.3. Paq](#_heading=h.vf93mmdrft1f) 15

[**SUBPROGRAMAS**](#_heading=h.pbwq92mz2slm) **15**

[7.1. Funciones](#_heading=h.eekgbns8f0ng) 15

[7.2. Procedimientos](#_heading=h.a1o81k8tg7i3) 16

[**CAPÍTULO II: GRAMÁTICA DEL LENGUAJE DE PROGRAMACIÓN**](#_heading=h.qag35co23hyd) **20**

[**CAPÍTULO III: ANALIZADOR SINTÁCTICO - ADP**](#_heading=h.6ernolk04eea) **22**

**INTRODUCCIÓN**

En el presente informe se propone un lenguaje creado para fines de los proyectos del curso de Lenguajes y Compiladores, se describe su estructura y la relaciones que tienen los elementos del lenguaje. Tiene como objetivo una programación con una sintaxis limpia, intuitiva y fácil de trabajar.

**CAPÍTULO I: DESCRIPCIÓN DEL LENGUAJE DE PROGRAMACIÓN**

1. **ESTRUCTURA DE UN PROGRAMA**

Nuestro lenguaje de programación está orientado a objetos de manera. En este punto veremos la estructura del programa.

|  |
| --- |
| Obvio clase Jugador [:: CLASE PRINCIPAL  :: ATRIBUTOS  :: Declaración de variables  entero edad^  palabra nombre^  :: declaración de constantes  DEFINIR real sueldo = 0^    :: SUBPROGRAMAS  Obvio vacuo correr (real cantidad) [  :: Sentencias  ]  Obvio vacuo saltar (real cantidad) [  :: Sentencias  ]  :: SUBPROGRAMA PRINCIPAL  Obvio vacuo CENTRAL () [  # SENTENCIAS  ]  ]  :: OTRAS CLASES  Obvio clase Club [  :: ATRIBUTOS  :: Declaración de variables  Obvio entero edad^  Obvio palabra nombre^    :: SUBPROGRAMAS  Obvio vacuo fichaje (entero cantidad) [  :: Sentencias  ]  Obvio vacuo contrato (entero cantidad)[  :: Sentencias  ]  Obvio real informacion () [  :: Sentencias  ]  ] |

***Nota:*** El operador “^” indica el término de cada sentencia en el lenguaje.

1. **IDENTIFICADORES**
   1. **Constantes**

Una constante es un valor que no puede ser alterado o modificado durante la ejecución de un programa, únicamente puede ser leído.

Las constantes se declararán con la directiva konstante seguida del tipo de variable , esto significa que esa constante tendrá el mismo valor a lo largo de todo el programa.

konstante tipo\_variable NOMBRE VALOR

|  |
| --- |
| konstante tipo\_variable NOMBRE VALOR |

|  |
| --- |
| :: Declaración de constantes  konstante entero Pi 3.1415926  CENTRAL( ) :: Función principal  [  :: Declaraciones locales de la función principal  entero a ^  :: Operación en función principal  a=Pi+5^  ] |

* 1. **Variables**

Una variable está formada por un espacio en el sistema de almacenaje (memoria principal de un ordenador) y un nombre simbólico (un identificador) que está asociado a dicho espacio. Además esto ayuda a poder organizarse mejor los tipos de datos a utilizarse.

|  |
| --- |
| tipo\_variable NOMBRE ^ |

|  |
| --- |
| CENTRAL( ) :: Función principal  [  :: Declaraciones de variables  entero a ^ :: números enteros  enteroG b ^ :: números enteros grandes  real d ^ :: números decimales  veracidad e ^ :: variable que guarda dos valores: Verdadero y Falso  caracter f ^ :: letra o símbolo  palabra g^ :: secuencia de símbolos o cadena de texto  ] |

* 1. **Vectores de una dimensión**

Se define al vector a una zona de memoria contigua que permite almacenar un conjunto de datos del mismo tipo. La cual se limitará mediante el valor escrito entre “[ ]”; si se colocaran los valores de este vector irán separados por comas.

|  |
| --- |
| tipo\_variable NOMBRE {numero\_de\_elementos}^ |

|  |
| --- |
| tipo\_variable NOMBRE {}==[valor1,valor2,valor3,…]^ |

|  |
| --- |
| CENTRAL( ) :: Función principal  [  # Declaracion de vectores unidimensional  entero a {3}^ :: vector unidimensional de 3 elementos  caracter b{}==[z,q,p,m] ^ :: vector unidimensional de 4 elementos  ] |

* 1. **Estructurados (Clases, atributos y métodos)**

Primero se debe definir los identificadores de acceso, por ejemplo: los accesibles des cualquier parte de la aplicación se utilizará “obvio”, los de solo accesible dentro de una clase “noobvio” y el que es accesible si esta dentro del mismo paquete y subclase “inmune”

Clases

Son el molde con el cual se representarán las entidades o conceptos.

Atributos

Son las características que tendrá nuestra clase (variables).

Métodos

Son las operaciones que puede hacer nuestra clase (procedimientos o funciones).

|  |
| --- |
| **modificador\_de\_acceso** clase **Nombre\_de\_clase [**  ::ATRIBUTOS  ::Declaración de variables  modificador\_de\_acceso tipo\_de\_dato Nomble\_variable^  ::METODOS  modificador\_de\_acceso tipo\_de\_variable\_de \_retorno NomMetodo (parámetros) [  ::Sentencias  retorna r^  ]  modificador\_de\_acceso vacuo NomMetodo (parámetros) [  ::Sentencias  ]  **]** |

1. **OPERADORES**

Nuestro lenguaje de programación, tiene varios operadores para realizar tareas que incluyen operaciones aritméticas, condicionales y lógicos.

Tiene una amplia gama de operadores para realizar diversas operaciones. Para una mejor comprensión de los operadores, estos operadores se clasifican como:

Los **operadores unarios** que son los que realizan una acción sobre un solo operando y los **operadores binarios** que realizan acciones con dos operandos y se asocian de izquierda a derecha.

* Aritméticos.- Los operadores aritméticos realizan operaciones matemáticas, como sumas o restas con operandos.

|  |  |
| --- | --- |
| Símbolo | Descripción |
| + | Sumar los dos operandos |
| - | Restar el segundo operando del primero |
| \* | Multiplicar los dos operandos |
| / | Dividir el primer operando por el segundo |
| \*\* | Elevar el primer operando a la potencia del segundo operando |
| // | Operador modulo. Obtiene el resto de dividir el operando de la izquierda por el de la derecha. |

* Asignación.- El operador de asignación se utiliza para asignar un valor a una variable. Este operador es el signo =.
* Incremento y decremento.- Se encuentran dos operadores que incrementan y decrementan el valor numérico de un operando (constante o variable) por 1(uno).

|  |  |
| --- | --- |
| Símbolo | Descripción |
| ++ | Incrementa el valor en 1 |
| -- | Decrementa el valor en 1 |

* Relacionales.- Se utilizan para comparar dos o más valores.

|  |  |
| --- | --- |
| Símbolo | Descripción |
| < | Menor que |
| > | Mayor que |
| == | Igual |
| ~= | No igual |
| <= | Menor o igual que |
| >= | Mayor o igual que |

* Lógicos.- Se usan para determinar la lógica entre variables o valores y estos devuelven Verdadero o Falso, dependiendo si la expresión es verdadera o falsa.

|  |  |
| --- | --- |
| Símbolo | Descripción |
| & | AND. Sólo será verdadero si todos los operadores son verdaderos. |
| | | OR. Será verdadero si sólo uno de los dos operadores es verdadero. |
| ~ | NOT |

1. **EXPRESIONES**

Las expresiones se definen como el conjunto de operadores de igual o distinta clase (descritos en el punto anterior) y operandos que pueden ser números o variables declaradas. En nuestro caso utilizaremos los paréntesis () para agrupar las expresiones.

Ejemplo:

|  |
| --- |
| CENTRAL( ) :: Función principal  [  :: Declaraciones locales de la función principal  entero a == 3^  entero b == 4^  :: Instrucciones de la función principal  suma(a,b)^  ] |

1. **INSTRUCCIONES**
   1. **Asignación**

La instrucción de asignación nos sirve para definir el valor de una variable  
En nuestro lenguaje, se utiliza la expresión “=”. En el caso de variables de tipo letra se pondrá comillas simples (‘’) alrededor del valor del carácter. En nuestro caso la variable declarada sea tipo cadena se usarán las comillas dobles (“”) para encerrar la cadena de caracteres.

Ejemplo:

|  |
| --- |
| entero edad =23^  caracter inicial = ‘a’^  palabra nombre = “Juan Perez” |

* 1. **Lectura / Escritura por pantalla**
* La instrucción de lectura de pantalla nos sirve para que el usuario pueda ingresar un dato por pantalla; este tiene que ser del mismo tipo de la variable establecida. Para ejecutar esta instrucción se debe usar el siguiente formato:

|  |
| --- |
| liyiy variable^ |

Ejemplo:

|  |
| --- |
| liyiy nombre^ liyiy promedio^ |

* La instrucción escritura nos permite mostrar en pantalla lo que el usuario desee a la hora de compilar el programa. Para hacerlo debe usar la siguiente instrucción:

|  |
| --- |
| iskirbiy variable^ o también iskirbiy “Mensaje a iskirbiy”^ |

En caso desee mostrar 2 o más variables o una combinación de variable o mensaje. Puede usar el “;” para concatenar términos.

Ejemplo:

|  |
| --- |
| iskirbiy nombre^  iskirbiy “Hola mundo”^  iskirbiy “Mi nombre es”; nombre; “apellido”^ |

1. **SENTENCIAS DE CONTROL DE FLUJO**

**6.1. Man - mana**

Son aquellas palabras reservadas que nos permite crear una estructura de control selectiva y es inseparable de una expresión o condición lógica. El “Si” nos permite ejecutar una acción en función a su condición y en el caso de no cumplirse el sino nos permitirá ejecutar otra acción.

|  |
| --- |
| man (expresión lógica)[  :: si la expresión es verdadera se ejecutará los siguientes sentencias  sentencias\_a  ]  mana [  :: si la expresión es falsa se ejecutará las siguientes sentencias  sentencias\_b  ] |

La estructura de control selectiva “sino” es útil para dar una mayor consistencia a la estructura condicional, el cual permite al usuario ejecutar otras instrucciones bajo la negación de la expresión lógica.

Ejemplo:

|  |
| --- |
| man (nota>=11) & (nota<=20)[    iskirbiy “Se aprobó el curso”^  ]  mana [  iskirbiy “No se aprobó el curso”^  ] |

**6.2. Kama**

Es una estructura iterativa o repetitiva que permite la ejecución de un bloque de sentencias si y solo si la condición o expresión lógica sea verdadera, de lo contrario finaliza el bucle.

|  |
| --- |
| kama (expresión lógica)[  sentencias\_a  ] |

Ejemplo:

|  |
| --- |
| kama (horas<=8)[  horas++^  ] |

**6.3. Paq**

Esta palabra reservada nos permite la elaboración de una estructura repetitiva definida ya que presenta un intervalo que hará posible el bucle de las sentencias hasta finalizar su ciclo debido a su incremento representado en la expresión lógica.

|  |
| --- |
| paq (inicialización, condición, incremento)[  sentencias\_a  ] |

|  |
| --- |
| paq (i==0, i<24, i++)[  iskirbiy “Horas del día”^  ] |

1. **SUBPROGRAMAS**

**7.1. Funciones**

Es un tipo de subprograma que está compuesto de un bloque de instrucciones, además utilizan parámetros cuyos valores nos sirven para efectuar operaciones y retornar un valor.

|  |
| --- |
| (modificador de acceso) (tipo de variable) NombreSubprograma (parámetros) [  sentencias\_a  retornar r^  ] |

*Nota:*El valor de retorno debe ser del mismo tipo de dato que se declara en la función.

|  |
| --- |
| Obvio real ValorFinal (real inicial, real tasa, real n ) [  retornar inicial\*(1 + tasa )\*\*^  ] |

Esta función retorna el interés compuesto mediante los parámetros formales inicial, tasa y el periodo. Para hacer el llamado a la función se escribe su nombre y los parámetros reales. El resultado se ejecuta y puede ser guardado en un dato de tipo real. Por ejemplo:

|  |
| --- |
| compuesto == ValorFinal (1500, 0.3, 7)^ |

**7.2. Procedimientos**

En contraste con las funciones, los procedimientos no retornan un valor y por ende utilizan un tipo de dato vacuo.

|  |
| --- |
| (modificador de acceso) vacuo NombreSubprograma (parámetros) [  sentencias  ] |

*Nota:* Recordar que la declaración del procedimiento vacuo hace referencia al tipo del valor de retorno, por lo que vacuo hace referencia al vacio.

|  |  |  |
| --- | --- | --- |
| **TOKEN** | **DESCRIPCIÓN** | **LEXEMA** |
| 101 | Operador | + |
| 102 | Operador | - |
| 103 | Operador | \* |
| 104 | Operador | / |
| 105 | Operador | \*\* |
| 106 | Operador | // |
| 107 | Operador | = |
| 108 | Operador | ++ |
| 109 | Operador | -- |
| 110 | Operador | < |
| 111 | Operador | > |
| 112 | Operador | == |
| 113 | Operador | ~= |
| 114 | Operador | <= |
| 115 | Operador | >= |
| 116 | Operador | & |
| 117 | Operador | | |
| 118 | Operador | ~ |

**CAPÍTULO II: GRAMÁTICA DEL LENGUAJE DE PROGRAMACIÓN**

G={Vn, Vt, Clases, P}

Vn={Clases, Clase, A, modificador\_acceso, B, R, Declaraciones, Subprogramas, Subprograma, G, Declaración, C, Tipo, S, Valor, E, Lista\_Valores, Lista\_ID, F, P,

K, I, H, Ret\_metodo, J, M, Parametros, Sentencias, Sentencia, Sentencia\_simple,

Relacional, 16, expresion, 13, U, RS, Z, W, 12, expresión\_mat, 10, 9, TR, V, W, X,

Op\_mat, 11, inicialización, condic, iteración, 17, op\_rel, 18, 20, 14, op\_conj\_dis, op\_comp, T}

Vt={clase, ID, [, ], ,, ^, konstante, ==, NUMERO, ‘ID’, verdad, falso, entero, enteroG, real, carácter, palabra, veracidad, vacuo, CENTRAL, (, ), man, kama, paq, liyiy, iskirbiy, ., \*, -, +, %, /, ++, --, =, ~=, &, |, obvio, noobvio, imnune}

P={

1. <Clases> --> <Clase><A>

2. <A> --> <Clase><A> | λ

3. <Clase> --> <modificador\_acceso> clase ID [<B>]

4. <B> --> <modificador\_acceso> <R>

5. <R> --> <Declaraciones> <Subprogramas> | <Subprograma> <G>

6. <Declaraciones> --> <Declaración> <C>

7. <C> --> ,<Declaraciones> | ^

8. <Declaración> --> <modificador\_acceso> <Tipo> ID <S> | konstante <Tipo> ID == <Valor>

9. <S> --> == <Valor> | λ | [<E>

10. <E> --> ] | <Lista\_Valores>]

11. <Valor> --> NUMERO | "<Lista\_ID>" | 'ID' | verdad | falso

12. <Lista\_Valores> --> <Valor> <F>

13. <F> --> , <Lista\_Valores> | λ

14. <Tipo> --> entero | enteroG | rea| | caracter | palabra | veracidad

15. <Subprogramas> --> <Subprograma> <G>

16. <G> --> <Subprograma> <G> | λ

17. <Subprograma> --> vacuo <P> | ID ( <K> ) [<I>] | <Tipo> ID (<K>) [<H> <Ret\_metodo>]

18. <P> --> CENTRAL () [<H>] | ID (<K>) [<H>]

19. <H> --> <I> <Sentencias>

20. <K> --> <tipo> ID <J> | λ

21. <I> --> <tipo> ID <M> | λ

22. <J> --> , <Parametros> | λ

23. <Parametros> --> <tipo> ID <J>

24. <M> --> ^ <I>

25. <Sentencias> --> <Sentencia> <Q>

26. <Q> --> <Sentencia> <Q> | λ

27. <Sentencia> --> <Sentencia\_simple>^ | man (<Relacional>) [<sentencias>] <16> | kama (<Relacional>) [<sentencias>] | paq (<expresion>) [<sentencias>]

28. <Sentencia\_simple> --> ID <13> | liyiy ID <U> | iskirbiy <RS>

29. <13> --> == <Z> | .ID(<W>)

30. <Z> --> ID <12> | (<expresion\_mat>)<12> | NUMERO<12> | "<Lista\_ID>" | 'ID' | verdad | falso

31. <12> --> <10> <9> | λ

32. <9> --> ID <12> | (<expresion\_mat>)<12> | NUMERO<12> | "<Lista\_ID>" | 'ID' | verdad | falso | λ

33. <U> --> [] | λ

34. <RS> -> ID <TR> | "<Lista\_ID>" <TR>

35. <TR> -> , <RS> | λ

36. <Lista\_ID> --> ID <V>

37. <V> --> ID <V> | λ

38. <W> --> ID <X> | <Valor> <X> | λ

39. <X> --> , <W> | λ

40. <expresion\_mat> --> NUMERO <10> | ID <10>

41. <10> --> <Op\_mat> <11> | λ

42. <11> --> ID | NUMERO

43. <Op\_mat> --> \* | - | + | % | /

44. <16> --> mana [<sentencias>] | λ

45. <expresion> --> <inicializacion>, <condic>, <iteracion>

46. <inicializacion> --> entero ID == NUMERO | real ID == NUMERO | ID <17>

47. <17> --> == NUMERO | λ

48. <condic> --> ID <op\_rel> <11>

49. <iteracion> --> ID <18>

50. <18> --> ++ | --

51. <Relacional> --> <20><14>

52. <14> --> <op\_conj\_dis><20><14> | λ

53. <op\_rel> --> = | < <op\_comp>| > <op\_comp>| ~=

54. <op\_comp> --> = | λ

55. <op\_conj\_dis> --> & | |

56. <20> --> <11> <op\_rel> <11>

57. <Ret\_metodo> --> retorna <T>

58. <T> --> ID<10> | "<Lista\_ID>" | 'ID' | verdad | falso | NUMERO<10>

59. <modificador\_acceso> --> obvio | noobvio | inmune

}

**CAPÍTULO III: ANALIZADOR SINTÁCTICO - ADP**

M[0][1] = "clase";

M[0][2] = "^";

M[0][3] = ",";

M[0][4] = "$";

M[0][5] = "konstante";

M[0][6] = "ID";

M[0][7] = "=";

M[0][8] = "&";

M[0][9] = "|";

M[0][10] = "[";

M[0][11] = "]";

M[0][12] = "NUMERO";

M[0][13] = comilla;

M[0][14] = "'";

M[0][15] = "falso";

M[0][16] = "verdad";

M[0][17] = "entero";

M[0][18] = "real";

M[0][19] = "caracter";

M[0][20] = "palabra";

M[0][21] = "veracidad";

M[0][22] = "vacuo";

M[0][23] = "{";

M[0][24] = "}";

M[0][25] = "#";

M[0][26] = "@";

M[0][27] = "CENTRAL";

M[0][28] = ".";

M[0][29] = "(";

M[0][30] = ")";

M[0][31] = "liyiy";

M[0][32] = "iskirbiy";

M[0][33] = ";";

M[0][34] = "\*";

M[0][35] = "-";

M[0][36] = "+";

M[0][37] = "%";

M[0][38] = "/";

M[0][39] = "man";

M[0][40] = "mana";

M[0][41] = "kama";

M[0][42] = "paq";

M[0][43] = ">";

M[0][44] = "<";

M[0][45] = "~";

M[0][46] = "retorna";

M[0][47] = "obvio";

M[0][48] = "noobvio";

M[0][49] = "inmune";

M[0][50] = "==";

//no hay forma de llegar a clase

M[1][0] = "<Clases>";

M[1][47] = "<A> <Clase>";

M[1][48] = "<A> <Clase>";

M[1][49] = "<A> <Clase>";

M[2][0] = "<Clase>";

M[2][47] = "} <B> { ID clase <modificador\_acceso>";

M[2][48] = "} <B> { ID clase <modificador\_acceso>";

M[2][49] = "} <B> { ID clase <modificador\_acceso>";

M[3][0] = "<A>";

M[3][2] = "λ";

M[3][47] = "<A> <Clases>";

M[3][48] = "<A> <Clases>";

M[3][49] = "<A> <Clases>";

M[4][0] = "<B>";

M[4][47] = "<R> <modificador\_acceso>";

M[4][48] = "<R> <modificador\_acceso>";

M[4][49] = "<R> <modificador\_acceso>";

M[5][0] = "<R>";

M[5][5] = "<Subprogramas> <Declaraciones>";

M[5][6] = "<G> <Subprograma>";

M[5][17] = "<G> <Subprograma>";

M[5][18] = "<G> <Subprograma>";

M[5][19] = "<G> <Subprograma>";

M[5][20] = "<G> <Subprograma>";

M[5][21] = "<G> <Subprograma>";

M[5][22] = "<G> <Subprograma>";

M[5][47] = "<Subprogramas> <Declaraciones>";

M[5][48] = "<Subprogramas> <Declaraciones>";

M[5][49] = "<Subprogramas> <Declaraciones>";

M[6][0] = "<Declaraciones>";

M[6][5] = "<C> <Declaración>";

M[6][47] = "<C> <Declaración>";

M[6][48] = "<C> <Declaración>";

M[6][49] = "<C> <Declaración>";

M[7][0] = "<C>";

M[7][3] = "<Declaraciones> ,";

M[7][4] = "|";

M[8][0] = "<Declaracion>";

M[8][5] = "<Valor> == ID <tipo> konstante";

M[8][47] = "<S> ID <tipo> <modificador\_acceso>";

M[8][48] = "<S> ID <tipo> <modificador\_acceso>";

M[8][49] = "<S> ID <tipo> <modificador\_acceso>";

M[9][0] = "<S>";

M[9][3] = "λ";

M[9][4] = "λ";

M[9][10] = "<E> [";

M[9][50] = "<Valor> ==";

M[10][0] = "<E>";

M[10][11] = "]";

M[10][12] = "] <Lista\_Valores>";

M[10][13] = "] <Lista\_Valores>";

M[10][14] = "] <Lista\_Valores>";

M[10][15] = "] <Lista\_Valores>";

M[10][16] = "] <Lista\_Valores>";

M[11][0] = "<Valor>";

M[11][12] = "NUMERO";

M[11][13] = comilla + "<Lista\_ID>" + comilla;

M[11][14] = "' ID '";

M[11][15] = "falso";

M[11][16] = "verdad";

M[12][0] = "<Lista\_Valores>";

M[12][12] = "<F> <Valor>";

M[12][13] = "<F> <Valor>";

M[12][14] = "<F> <Valor>";

M[12][15] = "<F> <Valor>";

M[12][16] = "<F> <Valor>";

M[13][0] = "<tipo>";

M[13][17] = "entero";

M[13][18] = "real";

M[13][19] = "caracter";

M[13][20] = "palabra";

M[13][21] = "veracidad";

M[14][0] = "<F>";

M[14][3] = "<Lista\_Valores> ,";

M[14][11] = "λ";

M[15][0] = "<Subprogramas>";

M[15][6] = "<G> <Subprograma>";

M[15][17] = "<G> <Subprograma>";

M[15][18] = "<G> <Subprograma>";

M[15][19] = "<G> <Subprograma>";

M[15][20] = "<G> <Subprograma>";

M[15][21] = "<G> <Subprograma>";

M[15][22] = "<G> <Subprograma>";

M[16][0] = "<G>";

M[16][6] = "<G> <Subprograma>";

M[16][17] = "<G> <Subprograma>";

M[16][18] = "<G> <Subprograma>";

M[16][19] = "<G> <Subprograma>";

M[16][20] = "<G> <Subprograma>";

M[16][21] = "<G> <Subprograma>";

M[16][22] = "<G> <Subprograma>";

M[16][24] = "λ";

M[17][0] = "<Subprograma>";

M[17][6] = "} <I> { @ <K> # ID";

M[17][17] = "} <H><Ret\_metodo> <H> { @ <K> # ID <tipo>";

M[17][18] = "} <H><Ret\_metodo> <H> { @ <K> # ID <tipo>";

M[17][19] = "} <H><Ret\_metodo> <H> { @ <K> # ID <tipo>";

M[17][20] = "} <H><Ret\_metodo> <H> { @ <K> # ID <tipo>";

M[17][21] = "} <H><Ret\_metodo> <H> { @ <K> # ID <tipo>";

M[17][22] = "<P> vacuo";

M[18][0] = "<P>";

M[18][6] = "} <H> { @ <K> # ID";

M[18][27] = "} <H> { @ # CENTRAL";

M[19][0] = "<H>";

M[19][17] = "<Sentencias> <I>";

M[19][18] = "<Sentencias> <I>";

M[19][19] = "<Sentencias> <I>";

M[19][20] = "<Sentencias> <I>";

M[19][21] = "<Sentencias> <I>";

M[19][24] = "<Sentencias> <I>";

M[19][46] = "<Sentencias> <I>";

M[20][0] = "<K>";

M[20][17] = "<J> ID <tipo>";

M[20][18] = "<J> ID <tipo>";

M[20][19] = "<J> ID <tipo>";

M[20][20] = "<J> ID <tipo>";

M[20][21] = "<J> ID <tipo>";

M[20][26] = "λ";

M[21][0] = "<I>";

M[21][6] = "λ";

M[21][17] = "<M> ID <tipo>";

M[21][18] = "<M> ID <tipo>";

M[21][19] = "<M> ID <tipo>";

M[21][20] = "<M> ID <tipo>";

M[21][21] = "<M> ID <tipo>";

M[21][24] = "λ";

M[21][31] = "λ";

M[21][32] = "λ";

M[21][39] = "λ";

M[21][41] = "λ";

M[21][42] = "λ";

M[22][0] = "<Parametro>";

M[22][17] = "<J> ID <tipo>";

M[22][18] = "<J> ID <tipo>";

M[22][19] = "<J> ID <tipo>";

M[21][20] = "<J> ID <tipo>";

M[22][21] = "<J> ID <tipo>";

M[23][0] = "<J>";

M[23][3] = "<Parametro> ,";

M[23][26] = "λ";

M[24][0] = "<M>";

M[24][4] = "<I> |";

M[25][0] = "<Sentencias>";

M[25][6] = "<Q> <Sentencia>";

M[25][31] = "<Q> <Sentencia>";

M[25][32] = "<Q> <Sentencia>";

M[25][39] = "<Q> <Sentencia>";

M[25][41] = "<Q> <Sentencia>";

M[25][42] = "<Q> <Sentencia>";

M[26][0] = "<Q>";

M[26][6] = "<Q> <Sentencia>";

M[26][24] = "λ";

M[26][31] = "<Q> <Sentencia>";

M[26][32] = "<Q> <Sentencia>";

M[26][39] = "<Q> <Sentencia>";

M[26][41] = "<Q> <Sentencia>";

M[26][42] = "<Q> <Sentencia>";

M[26][46] = "λ";

M[27][0] = "<Sentencia>";

M[27][6] = "| <Sentencia\_simple>";

M[27][31] = "| <Sentencia\_simple>";

M[27][32] = "| <Sentencia\_simple>";

M[27][39] = "man(<Relacional>){<sentencias>}<16> ";

M[27][41] = "kama(<Relacional>){<sentencias>}";

M[27][42] = "paq(<expresion>){<sentencias>}";

M[28][0] = "<Sentencia\_simple>";

M[28][6] = "<13> ID";

M[28][31] = "<U> ID liyiy";

M[28][32] = "<RS> iskirbiy";

M[29][0] = "<13>";

M[29][28] = "@ <W> # ID .";

M[29][50] = "<Z> ==";

M[30][0] = "<Z>";

M[30][6] = "<12> ID";

M[30][12] = "<12> NUMERO";

M[30][13] = comilla + "<Lista\_ID>" + comilla;

M[30][14] = "' ID '";

M[30][15] = "falso";

M[30][16] = "verdad";

M[30][29] = "<12> ) <expresion\_mat> (";

M[31][0] = "<12>";

M[31][4] = "λ";

M[31][34] = "<9> <10>";

M[31][35] = "<9> <10>";

M[31][36] = "<9> <10>";

M[31][37] = "<9> <10>";

M[31][38] = "<9> <10>";

M[32][0] = "<9>";

M[32][4] = "λ";

M[32][6] = "<12> ID";

M[32][12] = " <12> NUMERO ";

M[32][13] = comilla + "<Lista\_ID>" + comilla;

M[32][14] = " ' ID '";

M[32][15] = "falso ";

M[32][16] = "verdad";

M[32][29] = "<12> ) <expresion\_mat> (";

M[33][0] = "<U>";

M[33][4] = "λ";

M[32][10] = "] [";

M[34][0] = "<RS>";

M[34][6] = "<TR> ID";

M[34][13] = "<TR> <Lista\_ID>";

M[35][0] = "<TR>";

M[35][4] = "λ";

M[35][33] = "<RS> ;";

M[36][0] = "<Lista\_ID>";

M[36][6] = "<V> ID";

M[37][0] = "<V>";

M[37][6] = "ID";

M[37][13] = "λ";

M[38][0] = "<W>";

M[38][6] = "<X> ID";

M[38][12] = "<X> <valor>";

M[38][13] = "<X> <valor>";

M[38][14] = "<X> <valor>";

M[38][15] = "<X> <valor>";

M[38][16] = "<X> <valor>";

M[38][26] = "λ";

M[39][0] = "<X>";

M[39][3] = "<W> ,";

M[39][26] = "λ";

M[40][0] = "<expresion\_mat>";

M[40][6] = "<10> ID";

M[40][12] = "<10> NUMERO";

M[41][0] = "<10>";

M[41][4] = "λ";

M[41][6] = "λ";

M[41][12] = "λ";

M[41][13] = "λ";

M[41][14] = "λ";

M[41][15] = "λ";

M[41][16] = "λ";

M[41][24] = "λ";

M[41][29] = "λ";

M[41][30] = "λ";

M[41][34] = "<11> <Op\_mat>";

M[41][35] = "<11> <Op\_mat>";

M[41][36] = "<11> <Op\_mat>";

M[41][37] = "<11> <Op\_mat>";

M[41][38] = "<11> <Op\_mat>";

M[42][0] = "<11>";

M[42][6] = "ID";

M[42][12] = "NUMERO";

M[43][0] = "<Op\_mat>";

M[43][34] = "\*";

M[43][35] = "-";

M[43][36] = "+";

M[43][37] = "%";

M[43][38] = "/";

M[44][0] = "<16>";

M[44][6] = "λ";

M[44][24] = "λ";

M[44][31] = "λ";

M[44][32] = "λ";

M[44][39] = "λ";

M[44][40] = "} <sentencias> { mana";

M[44][41] = "λ";

M[44][42] = "λ";

M[44][46] = "λ";

M[45][0] = "<expresion>";

M[45][6] = "<iteracion> , <condic> , <inicializacion>";

M[45][17] = "<iteracion> , <condic> , <inicializacion>";

M[45][18] = "<iteracion> , <condic> , <inicializacion>";

M[46][0] = "<inicializacion>";

M[46][6] = "<17> ID";

M[46][17] = "NUMERO == ID entero";

M[46][18] = "NUMERO == ID real";

M[47][0] = "<17>";

M[47][3] = "λ";

M[47][50] = "NUMERO ==";

M[48][0] = "<iteracion>";

M[48][6] = "<18> ID";

M[49][0] = "<18>";

M[49][35] = "- -";

M[49][36] = "+ +";

M[50][0] = "<Relacional>";

M[50][6] = "<14> <20>";

M[50][12] = "<14> <20>";

M[51][0] = "<14>";

M[51][8] = "&";

M[51][9] = "|";

M[51][30] = "λ";

M[52][0] = "<20>";

M[52][6] = "<11> <op\_rel> <11>";

M[52][12] = "<11> <op\_rel> <11>";

M[53][0] = "<op\_rel>";

M[53][7] = "=";

M[53][43] = "<op\_comp> >";

M[53][44] = "<op\_comp> <";

M[53][45] = "= ~";

M[54][0] = "<op\_comp>";

M[54][6] = "λ";

M[54][7] = "=";

M[54][12] = "λ";

M[55][0] = "<op\_conj\_dis>";

M[55][8] = "&";

M[55][9] = "|";

M[56][0] = "<Ret\_metodo>";

M[56][46] = "<T> retorna";

M[57][0] = "<T>";

M[57][6] = "<10> ID";

M[57][12] = "<10> NUMERO";

M[57][13] = comilla + "<Lista\_ID>" + comilla;

M[57][14] = "' ID '";

M[57][15] = "falso";

M[57][16] = "verdad";

M[58][0] = "<modificador\_acceso>";

M[58][47] = "obvio";

M[58][48] = "noobvio";

M[58][49] = "imnune";

M[59][0] = "<condic>";

M[59][6] = "<11> <op\_rel> ID";

# **ANEXOS**



