**PROYECTO: ANALIZADOR SINTÁCTICO**

|  |
| --- |
| **Desarrollo:** Individual.  **Fecha de Presentación:** Enviar la carpeta del Proyecto al correo [ginobarroso@uagrm.edu.bo](mailto:ginobarroso@uagrm.edu.bo), con Asunto: *PROY-COMP. Nombre del alumno (e.g. PROY-COMP. Arce Catacora, Luis),* hasta el jueves 28/julio/2022, 23:59:59 horas. |

**Escribir una BNF** y luego **desarrollar un Analizador Sintáctico** (Parser) para el lenguaje MiniPASCAL, tomando en cuenta que el mismo **NO** es case-sensitive. Un Programa Pascal, consta de 3 secciones:

|  |  |
| --- | --- |
| 1) HEADER (el programador la puede obviar)  //Header → PROGRAM ID; | λ | PROGRAM Factorial; |
| 2) CUERPO: Mezcla de n ≥0 DECLARACIONES y PROCEDIMIENTOS  (Esta sección puede ser vacía)  //Cuerpo → … | VAR *//Declaración de* n ≥ 1 *líneas.*  a, b, c : INTEGER;  c, d : BOOLEAN;  PROCEDURE Algo; *//Los procedimientos no tienen parámetros,*  BEGIN *//ni variables locales.*  Sentencias;  *//El bloque puede estar vacío (λ)*  END; |
| 3) MAIN  (Sección obligada. Un programa si o si, debe tener al menos esta sección)  //Main -> BEGIN … END. | BEGIN  Sentencias; *//El bloque puede estar vacío (λ)*  END. *//Note que el END termina en PUNTO, no en PTOCOMA.* |

**SENTENCIAS DEL LENGUAJE MiniPASCAL**

Las sentencias del lenguaje son 8: Asignación, Llamada, Condicional, BucleFor, BucleWhile, BucleRepeat, Lectura, Impresión.

// Sentencia → Asignación | Llamada | Condicional | BucleFor | BucleWhile | BucleRepeat | Lectura | Impresión

*Como se sabe, las construcciones de programación: Condicional (IF-THEN-ELSE) y los Bucles WHILE y FOR, pueden tener una sola sentencia o un bloque BEGIN END; de sentencias. Recuerde que los bloques BEGIN END, pueden ser vacíos.*

|  |  |
| --- | --- |
| **ASIGNACIÓN.**  Se refiere a la asignación de una Expr (Expresión aritmética) a una variable.  //Asignación → ID **:=** Expr**;** | altura **:=** 25\*Base + z\*y; *//* **:=** *es el token ASSIGN* |
| **LLAMADA**  Para llamar a un procedimiento  //Llamada → ID( ); | factorial();  mostrar(); | |
| **CONDICIONAL.**  Se refiere a las construcciones IF-THEN e IF-THEN-ELSE  //Condicional → IF ExprBoole …  *Recuerde una regla de PASCAL: “Antes de un ELSE, no se escribe un punto y coma”.* | |  |  | | --- | --- | | IF z=3\*y and x+1<50  THEN  Println(“true”); | IF z>=0 OR (p+1< 0)  THEN  BEGIN  END  ELSE  BEGIN  READLN(x, y);  END; | | |
| **BucleFor**  Se refiere al bucle FOR de PASCAL en sus dos variantes: Una que usa TO y otra que usa DOWNTO  //BucleFor → FOR ID:= Expr TO Expr DO … | |  |  | | --- | --- | | FOR i:=1 **TO** n+1 DO  Println(“i=”, i); | FOR z:=2\*n **DOWNTO** n+1 DO  BEGIN  Println(“z\*2=”, z\*2);  READLN(p, q, s);  END; | | |
| **BucleWhile**  //BucleWhile → While ExprBoole DO … | |  |  | | --- | --- | | WHILE z=3\*y and x+1<50 DO  Println(“Infinito”); | WHILE z <= 2\*n DO  BEGIN  Println(z);  READLN(p, q, s);  z := z+1;  END; | | |
| **BucleRepeat**  Se refiere a la construcción Repeat-Until ExprBoole;  (El REPEAT-UNTIL puede estar vacío)  Este bucle NO usa el bloque BEGIN-END;  //BucleRepeat → REPEAT … UNTIL ExprBoole; | |  |  | | --- | --- | | REPEAT  Println(“Infinito”);  z := z-1;  UNTIL z < 0; | REPEAT  UNTIL p-1 < z\*3-5; | | |
| **LECTURA**  READLN( Uno o más ID’s separados con comas );  //Lectura → READLN( ID …); | |  |  | | --- | --- | | READLN(Altura); | READLN(a, b, c); | | |
| **IMPRESION**  PRINTLN(Mezcla de n ≥ 1 STRINGctte y Expr, separados con comas );  //Impresión → PRINTLN(…); | |  |  | | --- | --- | | WRITELN(“Hola Mundo”);  WRITELN(2\*i-5);  WRITELN(“Hola”, 2\*10-5); | WRITELN(“Hola”, “Mundo”, z+3, a/20, “bye”, (3\*i+2) MOD z); | | |

**APÉNDICE**

La producción para expresiones booleanas, ExprBoole, la definimos así:

ExprBoole → ExprBoole OR TermBoole | TermBoole

TermBoole → TermBoole AND FactorBoole | FactorBoole

FactorBoole → Expr OPREL Expr | (ExprBoole) | NOT FactorBoole //Expr = Expresiones aritméticas

***Nota. -*** *Esta definición de ExprBoole, no es completa (Faltan: ID, TRUE y FALSE). Si la escribimos en forma completa, genera ambigüedad.*

La producción para expresiones Aritmética, Expr:

Expr → Expr + Termino | Expr – Termino | Termino

Termino → Termino \* Factor | Termino / Factor | Termino MOD Factor | Factor

Factor → ID | NUM | - Factor | +Factor | (Expr)

***Nota2.-*** *En la definición de ExprBoole, en su producción de:*

FactorBoole → Expr OPREL Expr | (ExprBoole) | NOT FactorBoole

Las secciones Expr OPREL Expr | (ExprBoole) genera ambigüedad, por lo que se decidió quitar la sección (ExprBoole), entonces la producción quedaría así:

FactorBoole → Expr OPREL Expr | NOT FactorBoole