Descripción semántica

Definición de la estructura del programa

<PROG>::=Programa<NOMBRE>;[<LIBRERIAS>][<CONSTANTES>][<DEF_TIPOS>]
{<FUNC>}[<VARS>]<CUERPO>

Definición de nuestro programa, el cual tendrá un nombre y un cuerpo, además permitirá incluir librerías, declaración de constantes, definición de tipos, funciones y variables.

<NOMBRE>::=<LETRA>{<LETRA>|<DIGITO>}

<LET_MAY>::=A|..|Z

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

<DIGITO>::==0|..|9

Definición de las librerías

<LIBRERIAS>::=Usar <NOMBRE>{, <NOMBRE>};

Ejemplo:
Usar pantalla, matemáticas;

Definición de las constantes

<CONSTANTES>::=Const <CONS>{ <CONS>}

<CONS>::=<NOM_CONS> = <VAL_CONS>;

<NOM_CONS>::=<LET_MAY>{<LET_MAY>}

<VAL_CONS>::=VERDADERO|FALSO|<NUMERO>

<NUMERO>::=<DIGITO>{<DIGITO>}

Los valores de las constantes, pueden ser de tipo booleano (verdadero o falso), o de tipo numéricos, de los cuales sólo consideramos los valores enteros.

Ejemplo:
Const
P=146;
H= FALSO;

Definición de tipos

<DEF_TIPOS>::=Tipo <NOMBRE>=<TIPO>;{<NOMBRE>=<TIPO>;}

<TIPO>::=entero|real|booleano|cadena|<NOMBRE>

Representamos como será la declaración de los tipos definidos por el usuario.

Ejemplos:
Tipo ent = entero;

Tipo contador = ent;

Definición de variables

<VARS>::=Var <DECL_VAR>;{<DECL_VAR>;}

<DECL_VAR>::=<NOMBRE>{, <NOMBRE>}:<TIPO>

Ejemplo:
Var i,j:entero; aux:real;

Definición de función

<FUNC>::=Funcion <NOMBRE>['('<CABECERA>')']:<TIPO> Inicio {<SENT>;} Fin;

<CABECERA>::=[VAR] <DECL_VAR>{, [VAR] <DECL_VAR>}

En estas líneas definimos la forma de declarar funciones en nuestro lenguaje. Las funciones tendrán un nombre, y opcionalmente poseerán una cabecera, según si se le pasan parámetros a la función o no. Después aparecerá entre las palabras inicio y fin, el cuerpo de la función.

Ejemplo:
```
Funcion multiplica (num1: entero, num2: entero): entero
Inicio
        multiplica:= num1 * num2;
Fin;
```

Definición del cuerpo

<CUERPO>::=Inicio {<SENT>;} Fin.

<SENT>::=<ASIGNACION>|<ENTRADA>|<SALIDA>|<EXPR>|<INSERTAR_C>|
<BORRAR_C> |<BUCLE>|<CONDICION>

Las sentencias que podremos utilizar en nuestro lenguaje pueden ser asignaciones, leer valores de entrada por teclado, mostrar valores de salida por pantalla, expresiones, y también hemos considerado poner las operaciones de insertar_cadena y borrar_cadena porque son las dos operaciones del TDA cadena que no devuelven nada.

<ASIGNACION>::=<NOMBRE>:=<EXPR>

A un nombre se le asigna una expresión.
Ejemplo:
auxiliar:=4;

<ENTRADA>::= Lee <NOMBRE>

Lectura de valores.
Ejemplo:
Lee numero;

<SALIDA>::= Escribe '(' (<CADTEXTO> | <EXPR>) {,(<CADTEXTO> | <EXPR>)} ')'

Visualización a través de la salida estándar (pantalla).
Ejemplo:
*Escribe ('El resultado de ', num1,' * ', num2,' = ',multiplica(num1,num2));*

<CADTEXTO>::= '''{<Cualquier carácter>}'''

<EXPR>::=<EXPR>+<EXPR>|<EXPR>-<EXPR>|<EXPR>*<EXPR>|<EXPR>/<EXPR>|
<EXPR>Y<EXPR>|<EXPR>O<EXPR>|<EXPR>'>'=<EXPR>|<EXPR>'<'=<EXPR>|
<EXPR>'>'<EXPR>|<EXPR>'<'<EXPR>|<EXPR>'<''>'<EXPR>|<EXPR>=<EXPR>|
-<EXPR>|+<EXPR>|NO<EXPR>|
VERDADERO|FALSO|<LLAMADA_FUNCION>|'('<EXPR>')'|<NUMERO>|
<NOMBRE>|<BUSCA_C>|<EXTRAE_C>|<CONCAT_C>|<LONG_C>|<CADTEXTO>

Definimos todos los tipos de expresiones que soporta nuestro lenguaje.

<LLAMADA_FUNCION>::=<NOMBRE>['(' <EXPR>{, <EXPR>}')']

Definimos como serán implementadas las llamadas a las funciones. Cabe destacar que la función puede llevar o no expresiones en la cabecera, en caso de llevarlas irán separadas por comas.

Ejemplo:

```
devuelvePi;
factorial(5);
multiplica(num1, num2);
```

Definición del bucle Repite - Hasta

<BUCLE>::=Repite {<SENT>;} Hasta <EXPR>

Estructura del bucle Repite Hasta. Se repetirán una serie de sentencias hasta que se cumpla una expresión.

Ejemplo:

```
resultado:=cadenaVacia;
Repite
        buscaCancion(resultado);
Hasta resultado<>cadenaVacia;
```

Definición de la sentencia condicional

<CONDICION>::=Si <EXPR>
Entonces <BLOQUE_C>
[Sino <BLOQUE_C>]

Estructura que representa al if. Si se cumple una condición entonces se utilizarán una serie de sentencias, si no se cumple esa condición se utilizarán otras sentencias. Esta estructura permite el anidamiento.

Ejemplo:

```
Si sexo=='hombre' Entonces
        Escribe('Hola señor');
Sino
        Escribe('Hola señora');
```

<BLOQUE_C>::= <SENT> | Inicio {<SENT>;} Fin

Esta etiqueta nos permite etiquetas más claras en EBNF.

Definición de las operaciones asociadas al TDA Cadena

<INSERTAR_C>::=inserta'('<PARAM_CADENA>,<PARAM_CADENA>,<EXPR>')'

Inserta una subcadena en una cadena a partir de una posición inserta(cadena,subcadena,pos)

Ejemplo:

```
Cadena := 'hola '
```

Subcadena := 'mundo'
Pos:= 5
Inserta ('hola','mundo', 5);
{Resutado= 'hola mundo'}

<BORRAR_C>::=borra'('<PARAM_CADENA>,<EXPR>,<EXPR>')'

Borra en una cadena, a partir de una posicion P, N caracteres

Ejemplo:
Cadena:= 'pepe'
Posición:= 2

N_caractere:s= 1
borra('pepe',2,1); //llamada a la operación
{Resultado= 'pee'}

<BUSCA_C>::=busca'('<PARAM_CADENA>,<PARAM_CADENA>')'

Buscar en una cadena dada, una subcadena, devolviendo verdadero o falso, dependiendo de si la encuentra o no.

Ejemplo1:
Cadena:='hola mundo'
Subcadena:= 'mundo'
busca('hola mundo','mundo') //llamada a la operación
{Resultado= VERDADERO}

Ejemplo2:
Cadena:='hola mundo'
Subcadena:= 'adios'
busca('hola mundo','adios') //llamada a la operación
{Resultado= FALSO}

<EXTRAE_C>::=extrae'('<PARAM_CADENA>,<EXPR>,<EXPR>')'

Extrae N caracteres desde la posicion P en la cadena C.

Ejemplo:
Cadena:= 'hola mundo'
N_caracteres: = 2
Posición:= 2
extrae('hola mundo',2,2) //llamada a operación
{Resultado= 'la'}

<CONCAT_C>::=concatena'('<PARAM_CADENA>,<PARAM_CADENA>')'

Concatena dos cadenas pasadas como parámetros

Ejemplo:
Cadena1:='hola'
Cadena2:='chico'
concatena('hola','chico')
{Resultado= 'holachico'}

<LONG_C>::=longitud'('<PARAM_CADENA>')'

Devuelve la longitud de una cadena pasada como parámetro

Ejemplo:

Cadena:= 'hola'
longitud('hola') //llamada a la operación
{Resultado= 4}

<PARAM_CADENA>::=<NOMBRE>|<LLAMADA_FUNCION>|<CONCAT> | <EXTRAE_N_P> | <CADTEXTO>

Definimos los distintos tipos de parámetros que podrían contener las cabeceras de las operaciones del TDA cadena de caracteres.

<COMENTARIO>::='{'{<Cualquier carácter>}'}'

{Esto es un comentario en nuestro lenguaje}