Diseño de nuestra propuesta de lenguaje

Empty Stack

En este documento encontraras las partes necesarias para tener una propuesta de lenguaje mínima, la cual puede ser extensible hasta donde tu imaginación quiera para el desarrollo de un procesador de lenguaje, en este caso un compilador. Estas partes en rasgos generales son:

Características:

Este lenguaje permite a los usuarios realizar tareas y procesos sencillos con la finalidad de un rápido aprendizaje a la teoría de la programación orientada a objetos, la lógica de la programación y lo relacionado, como:

- Declaración de variables locales y globales
- Declaración de métodos
- Declaración de funciones
- Llamado y ejecución de funciones
- E/S de datos.

Alfabeto:

Abarca todas las letras del abecedario, desde la A a la Z, incluyendo las minúsculas, así como todos los dígitos:

[A,..,Z],[a,..,z],[0,1,..,9]

De igual forma, incluye todos los demás caracteres del código ASCII:

[>,<,=,<,+,*, &,%,#, /…]

De los anteriores mencionados, solo algunos caracteres tales como los operadores aritméticos, lógicos y relaciones junto con el carácter '%' o '//' tendrán una acción dentro del compilador, los demás solo serán caracteres reconocibles por el mismo, por ejemplo, en caso de querer imprimir una cadena.

Lenguaje:

Dentro del lenguaje las cadenas que reconoceremos serán identificadores y palabras reservadas por el sistema:

Identificadores

Todos los identificadores posibles generados por la siguiente expresión regular con la condición de que ningún identificador puede ser una palabra reservada:

[A-Z,a-z]+[A-Z,a-z,0-9]*

Palabras Reservadas

Las palabras reservadas las dividiremos por su función, acción o uso dentro de nuestro compilador, ya séase una estructura de control o incluso un operador aritmético abra métodos sobrecargados que nos permitan hacer la misma acción de maneras diferentes para poder facilitar la programación.

Estructuras de control

- SI - HAZ - ENTONCES

Evalúa una proposición y efectúa una acción si el resultado es verdadero, con el siguiente formato: si proposición=verdadera entonces ejecutar acción. De manera opcional, se puede establecer una acción a realizar para el caso de que la proposición evaluada resulte falsa: si proposición=verdadera entonces ejecutar acción 1. Si no, ejecutar acción 2.
Sintaxis:

```
//SI - HAZ - ENTONCES
SI (proposición=verdadera) HAZ{
ejecutar acción 1}
ENTONCES{
ejecutar acción 2}
```

- LOOP - A

realiza una iteración una cantidad de veces específica. Para ello necesita una variable declarada por el usuario, dicha variable debe ser de tipo entero, a la que asignará valores dentro del rango que se le especifique, incrementándose en uno. Por ejemplo, si se declara una variable llamada “i” se puede utilizar un LOOP con la siguiente sintaxis:

```
//LOOP - A
                    LOOP i = 5 a 9 cuando{
                                    sentencia}
```

Operadores Aritméticos

Son aquellos que sirven para operar términos numéricos. Estos operadores podemos clasificarlos a su vez como :

Unarios

Los operadores UNARIOS son aquellos que trabajan con UN OPERANDO. Este lenguaje permite el manejo de un operador unario llamado:

MENOS UNARIO

Este operador denota la negación del operando, y se representa por medio del signo menos (-) colocado antes del operando. Por ejemplo :

Si x tiene asignado el valor 100, -x dará como resultado -100 ; esto es que el resultado es el inverso aditivo del operando.

Binarios

Los operadores BINARIOS, son los que combinan DOS OPERANDOS , dando como resultado un valor numérico cuyo tipo será igual al mayor de los tipos que tengan los operandos

Operador	Acción
+	El operador + suma los valores de tipo de datos numéricos.
-	El operador - resta los valores de tipo de datos numéricos.
*	El operador * multiplica los valores de tipo de dato numérico.
^	El operador ^ calcula el exponente entre valores de tipo de datos numéricos.
/	El operador / calcula la división entre valores de tipo de datos numéricos.
%	El operador % no hace otra cosa que devolver el resto de la división entre los dos operandos

Operadores relacionales

disponemos de los operadores relacionales para verificar si se cumple una relación. Por ejemplo, el operador de equivalencia (==) nos devuelve un valor de verdadero si los operandos son iguales. Estas operaciones comparan dos valores numéricos y devuelven un valor booleano. Funcionamiento:

Operador	Proposición	Definición
>	A>B	Verdadero si A es mayor que B
>=	A>=B	Verdadero si A es mayor o igual que B
<	A<B	Verdadero si A es menor que B
<=	A<=B	Verdadero si A es menor o igual que B
==	A == B	verdadero si A es igual a B
!=	A != B	verdadero si A es diferente de B

Tipos de dato

INT: El tipo de dato entero es muy importante pues permite establecer variables que pueden almacenar datos enteros tanto positivos como negativos, el rango que se le atribuye a este tipo de dato es de -2,147,483,648 a 2,147,483,647 que equivale a -2^31 a 2^31

```
//Esto es una declaración de enteros

    INT VAR1 = 100;
```

DEC: Este tipo de dato nos permite crear variables que almacenen valores con números decimales ya sean positivos o negativos ocupa un rango comprendido de -3,4028235E+38 a 3,4028235E.

```
//Esto es una declaración de decimal

    DEC VAR2 = 1.1;
```

CAD: Con este tipo de dato podemos crear variables que almacenen cadenas de texto. Este tipo de dato tiene un rango positivo de 2^31. Para poder crear una cadena se tiene que capturar entre comillas simples o dobles el texto que se quiere transformar

```
//Esto es una declaración de cadenas

    CAD VAR4 = 'HOLA MUNDO';

    CAD VAR5 = "HOLA MUNDO";
```

Métodos

RET: Método que hace el retorno en una función de lo que obtenga en el siguiente token, este token siguiente tiene que ser una variable por ende un identificador declarado localmente en la función o globalmente en el archivo de código, esto nos ayuda a hacer saltos entre el código o mandar variables locales a funciones externas, simulando el almacenamiento global en la función.

```
//Método para el retorno de variables

   RET VAR1;
```

ETD: En el lenguaje de programación que estamos diseñando usamos ETD para captar la interrupción de teclado de mas de un carácter, así el programador puede pedir al usuario del código que ingrese datos desde teclado y almacenarlos en una variable, requiere de un uso correcto de los datos, en caso de capturar números en una variable de tipo CAD estos números tomaran el valor de CAD y no de ENT, en caso contrario (caracteres de tipo CAD en ENT) simplemente no se podrá hacer el almacenamiento de estos valores.

```
//Este es el método para la ENTRADA de datos, Proviene
de un acrónimo y aportación de ENTRADA = ETD.

     CAD VAR10 = ETD();
```

SLD: En el lenguaje de programación que estamos diseñando usamos SLD para mostrar en pantalla los datos almacenados en una variable, para esto ocupamos mandar a llamar el método SLD junto con RET, a SLD se le pasa como parámetro la variable o cadena a imprimir.

```
//Este es el método para la SALIDA De datos, proviene de
un acrónimo y aportación de SALIDA = SLD

     RET SLD(VAR10);

     RET SLD("HOLA MUNDO");

     RET SLD('HOLA MUNDO');
```

Estructuras del lenguaje de programación

//: Para realizar comentarios de una sola línea se utiliza la subcadena //, antes cada comentario para validar esta opción.

```
//Esto es un comentario en un solo renglón.
```