DOCUMENTACIÓN OFICIAL

By: Brun Juan &

Turin Barragan Brian Emanuel

Indice

Introducción al Trabajo Final

Desde la materia Sintaxis y Semántica de los Lenguajes se nos demandó la tarea de la elaboración de un interprete para un lenguaje de programación creado en base a la teoría dictada durante el cursado de esta materia.

Este lenguaje cumple con los objetivos básicos pedidos en la consigna de este mismo trabajo.

El objetivo de los creadores fue obtener un lenguaje que sea parecido a Pascal pero con algunas modificaciones típicas de otros lenguajes conocidos y además una traducción al español.

Integrantes:

BRUN, Juan.

TURIN BARRAGAN, Brian Emanuel.

¿Qué se puede hacer con “Pascaln’t”?

Pascaln’t satisface las siguientes sentencias:

- Asignación de un valor a una variable.
- Resolver operaciones matemáticas (suma, resta, multiplicación, división, potencia y raíz).
- Condicional, tanto para verdadero como para falso de la condición.
- Ciclo tipo “while” o “mientras”.
- Leer el valor de una variable a la vez que se imprime una cadena.
- Imprimir, pudiendo así hacerlo de una forma sucesiva de variables y/o cadenas y números.

Nota: las variables pueden ser del tipo real o bien arreglo de reales y son declaradas al principio.

¿Cómo programar en Pascaln’t?

Una vez entendemos que tipo de funcionalidades nos permite usar el lenguaje, nos interesará comprender cual es la forma correcta en la cuál el mismo reconocerá nuestra escritura.

Un programa es Pascaln’t comienza con la palabra reservada “**programita**”. Luego se permite (no es necesario) ingresar un título de la forma: “ **seLlama ‘<un_título>’;** ”.

Luego se permite declarar las variables, las mismas se deben ingresar una después de otra, separadas por coma (en caso de ser un vector reemplazar el nombre de la variable real por: "**<un_nombre> vector[<un_valor>]**"), de la siguiente forma.

"**definición <variable_1>,<variable_2>,...,<variable_n>;**"

Luego de la declaración de variables, se utiliza un par de llaves: "{" y "}" respectivamente para encerrar el cuerpo de sentencias de la forma:

{

<sentencia_1>;

<sentencia_2>;

.

.

.

<sentencia_n>

}

Observaciones:

La cantidad mínima de sentencias es 1 (una).

La última sentencia no lleva el separador ";" (punto y coma).

Tipos de Sentencia:

-Asignación:

Para realizar una asignación la sentencia debe tener la forma:

<variable> = <valor>; (variable real).

<variable>[<numero_componente>] = <valor>; (variable vector).

La variable puede ser una real o una componente de un vector.

El **<valor>** puede ser una constante escrita en el momento o el contenido de uan variable.

-Condición:

Para realizar una condición la sentencia debe tener la forma:

"**si <condición> { <sentencias> } sino { <sentencias> }**"

La condición debe devolver un valor booleano del tipo verdadero/falso. Admitiendo el uso de conjunción mediante el operador lógico "@" y disyunciones inclusivas con el operador lógico "**:**".

Para utilizar una condición, no es necesario escribir un cuerpo que funcione por falso de la misma, esto solo se utilizará si el programador lo cree necesario. Si solo se trabajará por verdadero del valor de verdad, tan solo debe considerar: "**si <condición> { <sentencias> }**".

- Leer:

Para realizar una lectura la sentencia debe tener la forma:

"**leer('<una_cadena_a_mostrar>',<una_variable>)**"

<una_cadena_a_mostrar> se imprimirá en pantalla y se utilizará para lo comunicación con el usuario sobre la información que debe proporcionar. Una vez el mismo ingrese el dato, el mismo se almacenará en la variable puesta en **<una_variable>**.

- Mostrar:

Para mostrar un valor por pantalla la sentencia debe tener la forma:

"**imprime('<una_cadena_a_mostrar>,<una_variable>)**"

<una_cadena_a_mostrar> se imprimirá en pantalla y se utilizará para lo comunicación con el usuario sobre la información que debe proporcionar. Luego, se imprimirá también en valor almacenado en <una_variable> a la hora de la ejecución de esta porción del código.

- Ciclo:

El único ciclo que se admite es de tipo "while" o "mientras", de la forma:

"**mientras <condición> {<sentencias>}**"

<condición> utilizará una variable de control en su comparación, la cual detendrá o continuará la ejecución del ciclo de manera conveniente para iterar o no el cuerpo de **<sentencias>**.

Advertencia: es MUY importante renovar el valor de la variable de control. De esta forma se llevará un control de la ejecución del ciclo, un error en este

diseño del programa puede generar un ciclo infinito que estropeará el funcionamiento del mismo.

Puesta en funcionamiento.

El intérprete está diseñado para trabajar con un archivo del tipo “.txt” que será donde se escribirá el lenguaje que se busca ejecutar.

Para usar el archivo que usted desee, basta con modificar la constante “rutaFuente” en el intérprete con la dirección del archivo “.txt” y ya funciona.

Ejemplo de programa simple:

El siguiente programa escrito con la sintaxis y semántica característica de “Pascaln´t” pide el ingreso de un número al usuario y luego lo imprime en pantalla.

```
programita
seLlama 'Ejemplo_1';
definicion num;
{
  leer('Ingrese un número',num);
  imprime('El número ingresado es: ',num)
}
```

El siguiente programa pide ingresar la cantidad de componentes de un vector, luego el valor de cada una de estas y luego cicla y los muestra.

```
programita
seLlama 'Ejemplo_2';
definicion cant,I,aux,v vector[50];
{leer('Ingrese la cantidad de componentes del vector',cant);
  si cant>0
  {
  I = 1;
```

```
    mientras I <= cant
    {
    leer('Ingrese componente',aux);
    v[I] = aux;
    I = I + 1
    };
    I = 1;
    mientras I <= cant
    {
    imprime('Componente numero',I);
    imprime(':',v[I]);
    I = I +1
    }
  }
}
```

Invitamos al lector a experimentar con las funcionalidades y ponerse en contacto con los creadores para mostrar los programas que se han podido diseñar con esta herramienta.

Simbología

Cadena: una cadena en Pascaln't se escribe como una concatenación sucesiva de 0 o más caracteres entre “”.

Ejemplo: ‘aaa’, ‘Hola’, ‘5782’.

Operadores Algebraicos:

+ : Define una suma de la forma <operador_1> + <operador_2>.

- : Define una resta de la forma <operador_1> - <operador_2>.

* : Define una multiplicación de la forma <operador_1> * <operador_2>.

/ : Define una división de la forma <operador_1> / <operador_2>.

La prioridad definida para los operadores algebraicos es:

Primero se resuelve la Potencia y Radicación, luego la Multiplicación y División y por último la Suma y Resta. Cada par de esta se priorizara dependiendo la posición en la expresión en la que se encuentran, siendo las que se resuelven primero aquellas que se encuentran más a la izquierda (se resuelve de izquierda a derecha).

Operadores Logicos:

: - Define a la disyunción inclusiva de la forma

<valor_verdad1> : <valor_verdad2>.

@ - Define a la conjunción de la forma

<valor_verdad1> @ <valor_verdad2>.

La prioridad definida para los operadores lógicos es:

Primero se resuelve la disyunción inclusiva y luego la conjunción. Los valores de verdad son el resultado de expresiones que involucran a operadores relacionales.

Operadores Relacionales:

== : Igual.

<> : Distinto.

> : Mayor que.

>= : Mayor o igual que.

< : Menor que.

<= : Menor o igual que.

Su funcionamiento no presenta ninguna diferencia con la forma clásica en la que se dedica su uso: <exp_algebraica1> <op_relacional> <exp_algebraica2>

Estructura

El programa que corre el intérprete se compone de 3 partes principales:

Analizador Léxico: el cual carácter a carácter clasifica las distintas palabras y además maneja las palabras reservadas en una Tabla de Símbolos.

Analizador Sintáctico: se compone de una pila y un árbol, además de la Tabla de Análisis Sintáctico (TAS). El mismo se encarga de verificar si las distintas cadenas forman producciones válidas y si la estructura del código es correcta a la forma en la que este está definido.

Evaluador/Analizador Semántico: compuesto por una gran lista de procedimientos evaluadores, se encarga de (una vez correctamente escrito) darle comportamiento al lenguaje. En el mismo se especifican las prioridades, que no es sensible a mayúsculas, se maneja un vector “Estado” que contiene las variables y su tipo y valor(es) asociado(s).

Opiniones.

Los creadores del lenguaje nos dejan plasmadas algunas impresiones sobre el mismo.

-Pascaln’t fue diseñado con el fin de culminar y poner en funcionamiento diversos temas que se dictan en la cátedra. Sé que posiblemente el lenguaje está bastante acotado en funcionalidad, además de que no permite la excelente comodidad a la cual otros lenguajes se adaptan y muestran.

En lo personal el proyecto nos trajo varios momentos duros, en los cuales tuvimos que darle vueltas una y otra vez a los distintos problemas que surgían para poder intentar ayudar. Sin embargo, una vez terminado, fue muy satisfactorio ver el programa en ejecución y corriendo el código escrito en el archivo Fuente.-