Universidad del Quindío

Programa Ingeniería de Sistemas y Computación

Asignatura Teoría de Lenguajes Formales

Diseñado realizado por: Leonardo Hernández

Editado y modificado por: Claudia Elena Quiceno Restrepo

Fase 2.1 del Proyecto de asignatura

Especificación formal de los tokens de un lenguaje de programación

**Semestre académico:** I semestre académico de 2015

**Profesor:** Claudia Elena Quiceno Restrepo.

## Objetivo

*Especificar los tokens de un lenguaje de programación indentificados por los estudiantes, mediante expresiones regulares y un autómata finito determinista*

## Convenciones

Para la especificación de los tokens del lenguaje, utilice las siguientes convenciones:

L: Letra mayúscula o minúscula.

M: Letra mayúscula

m: Letra minúscula

D: Dígito; 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

S: Cualquier símbolo excepto los delimitadores de cadena.

## Actividades

Realice las siguientes actividades en el formato anexo.

**1)** Escriba el nombre para el lenguaje de programación que va a codificar.

**2)** Se debe identificar las siguientes categorías léxicas y realizar los autómatas correspondientes, también para efectos de la aplicación y facilidad en la interfaz se podrá elaborar el diagrama de transición de un (solo 1) Autómata finito determinista que acepte todos los tokens de su lenguaje de programación. Puede omitir por simplicidad los ‘sumideros de rechazo’, que no prestan mayor utilidad en este tipo de proyectos:

1. Los identificadores de un lenguaje de programación, que sean usados para escribir un código fuente de una sola clase. (los identificadores más usados).
   1. Palabras reservadas
   2. Identificadores de variables
   3. Estándar para los identificadores de clase y método
   4. NOTA: no olvidar las palabras reservadas, para estructuras de selección y de control.
2. Los siguientes símbolos
   1. Operadores relacionales
   2. Operadores aritméticos
   3. Operadores lógicos
   4. Símbolo de asignación
   5. Carácter que signifique terminal o inicial, si lo contiene
   6. Cualquier otro símbolo (carácter significativo, que sea usado para algo en especial dentro del lenguaje a diferencia de java, ejemplo: ? , símbolo de php).
3. La forma en que se identifican los números enteros, reales, tipos binarios, cadena de caracteres, caracteres.
   1. Identificar los caracteres especiales, para los tipos anteriores.
4. Tipos de datos que manejan.

**3)** Escriba la expresión regular para cada uno de los tokens del lenguaje.

(Deben escribir las expresiones regulares al menos para los siguientes tipos de token:

1. Enteros
2. Reales
3. Identificadores: nombres de variable, nombres de función, nombres de clase, nombres de método, palabras reservadas, etc.
4. Operadores aditivos: más y menos
5. Operadores multiplicativos: por y dividido por
6. Símbolo de abrir
7. Símbolo de cerrar
8. Operador u operadores de asignación
9. Operadores relacionales
10. Operadores lógicos
11. Separadores de sentencias, si se requieren
12. Cadenas de caracteres
13. Otro tipo de token elegido por los estudiantes

**4)** Escriba las observaciones o cosas importantes que resaltar del lenguaje que están analizando:

Universidad del Quindío

Programa Ingeniería de Sistemas y Computación

Asignatura Teoría de Lenguajes Formales

Solución de la Fase 2.1 del Proyecto de asignatura

Especificación formal de los tokens de un nuevo lenguaje de programación

**Semestre académico:** I semestre académico de 2015

**Presentado por :**

Nombre: Jesica Tapasco Velez

Nombre: Santiago Montaño Lince

Nombre: Sebastian Romero

**Fecha:** 10/05/2015

**Presentado al profesor:** Claudia Elena Quiceno Restrepo.

***Desarrollo***

1. El lenguaje en el cual se trabajó fue PHP, el cual es un acrónimo de Hypertext Preprocessor aunque originalmente significaba Personal Home Page Tools.
2. Documento adjunto en el comprimido del proyecto, con nombre “AUTÓMATAS FINITOS DETERMINISTAS LENGUAJE PHP”.
3. Enteros

Decimal: [1-9][0-9]\*

| 0

Hexadecimal: 0[xX][0-9a-fA-F]+

Octal: 0[0-7]+

Binario: 0b[01]+

Reales: LNUM: [0-9]+

DNUM: ([0-9]\*[\.]{LNUM}) | ({LNUM}[\.][0-9]\*)

EXPONENT\_DNUM: [+-]?(({LNUM} | {DNUM}) [eE][+-]? {LNUM})

Identificadores: nombres de variable, nombres de función, nombres de clase, nombres de método, palabras reservadas, etc.

Nombres de variable: $[a-zA-Z\_\x7f-\xff][a-zA-Z0-9\_\x7f-\xff]\*

Nombres de función: [a-zA-Z\_\x7f-\xff][a-zA-Z0-9\_\x7f-\xff]\*

Nombres de clase: ^[a-zA-Z\_\x7f-\xff][a-zA-Z0-9\_\x7f-\xff]\*$

Palabras reservadas:

While: [w][h][i][l][e]

If: [i][f]

Do: [d][o]

Echo: [e][c][h][o]

Operadores de visibilidad: [p][(ublic) U r(ivate U rotected)]

Operadores aditivos: más y menos

[+ U -]

Operadores multiplicativos: por y dividido por

[\* U /]

Símbolo de abrir

[(]

Símbolo de cerrar

[)]

Apertura de bloque

[{]

Cierre de bloque

[}]

Operador u operadores de asignación

[=]

Operadores relacionales

[== U === U != U <> U !== U < U > U <= U >= U <=>]

Operadores lógicos

[AND U XOR U OR U ! U && U ||]

Cadenas de caracteres

“[D U L U \_ U S]”

1. PHP es un lenguaje de alto nivel, originalmente orientado hacia el desarrollo web de contenido dinámico. Aunque este también tiene orientación a objetos, es multiplataforma y es muy útil debido a su programación del lado del servidor que incorpora internamente el documento HTML. PHP tiene cualidades que permiten al desarrollador una mejor experiencia, debido a la gran cantidad de herramientas que trae consigo.

Una de las desventajas del lenguaje es que para algunos objetos, como los String, el memorizar sus funciones o métodos resulta complicado debido a que la notación varía mucho de uno a otro.

Ej:

Ejemplos de funciones de strings, sin guion bajo:

* strlen
* strrev
* htmlspecialchars

Ejemplos con guion bajo:

* str\_pad
* str\_shuffle
* get\_html\_translation\_table

Este otro caso también tenemos: tempnam vs tmpfile. En una usa el prefijo tmp y en la otra temp.

Y para denotar insensibilidad a mayúsculas/minúsculas: stripos (usa una i para denotar case-insensive), pero luego nos encontramos con strcasecmp (en este caso introduce case).

Lo que resulta en cierta complejidad cuando se está trabajando con las funciones de ciertos objetos.

Debido a que PHP no utiliza tipos de datos, suple la necesidad de comparación entre diferentes objetos con una amplia cantidad de operadores de comparación, esto quiere decir que, en el caso de tener dos variables con el valor de 4, pero una de estas entera y la otra String, tenemos un operador de comparación simplemente para el contenido (==) y otra que verifica completamente la igualdad (===) incluso teniendo en cuenta el tipo de dato, por lo tanto la comparación con == será true pero la comparación con === dará como resultado false.

Ej:

$varEntera = 4;

$varStrg = “4”;

$varEntera == $varStrg Daría como resultado True.

Pero…

$varEntera === $varStrg Daría como resultado False.

En PHP hay dos maneras de denotar una cadena de caracteres, las cuales son las comillas dobles y las comillas simples. La diferencia entre ambas obtiene relevancia al momento de realizar una concatenación o de expandir sentencias de escape para caracteres especiales.

Una de las diferencias entre el entrecomillado doble y simple en PHP es que en el entrecomillado simple es que las sentencias de escape para caracteres especiales no se expandirán.

Ej:

// Resultado: Ha borrado C:\\*.\*?  
echo 'Ha borrado C:\\*.\*?';  
  
// Resultado: Esto no se expandirá: \n una nueva línea  
echo 'Esto no se expandirá: \n una nueva línea';

En el caso de las comillas dobles esto si ocurrirá.

Ej:

$genial = 'fantástico';

// Ambos muestran: Esto es fantástico  
echo "Esto es {$genial}";  
echo "Esto es ${genial}";