Práctica 05

DOCENTE	CARRERA	CURSO
MSc. Vicente Enrique Machaca Arceda	Escuela Profesional de Ingeniería de Software	Compiladores

PRÁCTICA	TEMA	DURACIÓN
05	Especificación Léxica	3 horas

1. Datos de los estudiantes

- Grupo: 1
- Git Hub: `https://github.com/CrazyDani17/practica5_compiladores`
- Integrantes:
  - Guillermo Alemán
  - Marvik Del Carpio
  - Daniel Mendiguri
  - Daniela Vilchez

2. Competencias del curso

- Conocer las bases de la teoría de la computación para la implementación de un compilador.
- Implementar un compilador para un lenguaje de baja complejidad.

3. Competencias de la práctica

- Entender como convertir un autómata finito no determinista a un autómata finito determinista.

4. Equipos y materiales

- Latex
- Conección a internet

5. Entregables

- Se debe elaborar un informe en **Latex** donde se responda a cada ejercicio de la Sección 6.
- Por cada 5 minutos de retraso, el alumno tendrá un punto menos.

6. Ejercicios

En sus respectivos grupos, deben determinar/crear un nuevo lenguaje. Para esto, empezaremos definiendo la especificaión léxica. La especificación léxica es un documento donde se detalla como será cada componente léxico, en si debe contener:

1. Definición de los comentarios.

```
Comentarios en bloque: /* texto */
Comentarios en lnea: //texto
```

2. Definición de los identificadores.
   Los identificadores deben empezar con una letra, no está permitido que un identificador empiece con un número o un subguión.
   - Los caracteres de la “A” a la “Z”.
   - Los caracteres de la “a” a la “z”.
   - Números y subguión.
3. Definición de las palabras clave.
   int
   null
   string
   float
   char
   nullptr
   bool
   boolean
   char
   const
   func
   for
   if
   while
   true
   false
   return
   else
   elif
   class
4. Definición de los literales.
   - Literales enteros
     1
   - Literales entero flotante
     2.0
   - Literales booleanos
     true, false
   - Literales caracteres

```
nueva linea \n
tabulador \t
```

- Literales string

```
string vacio ""
string "asdasda"
```

5. Definición de los operadores.
   Los siguientes caracteres se utilizan en el código fuente como operadores:

```
  + - !  %  **  &  *  |  /  >  <
( )  {  }  [  ]  ;  ,  .  =
```

   Las siguientes combinaciones de caracteres se utilizan como operadores:

```
  + - ++ -- == <= >= !=
+= -= *= /= << >> || &&
```

6. Expresión regular de cada componente léxico (en una tabla).

Componente Léxico	**Expresión Regular**
Comentario en Bloque	\/\(\w|\W)*\/
Comentario en Línea	\/\/(...)*
Identificadores	[a-z|A-Z]+[\w]*
String Vacio	""
Literal Entero	0|[1-9][0-9]*
Literal Flotante	(0|[1-9][0-9])\.[0-9]
Literal Booleano	true|false
Nueva Linea	\n
Tabulador	\t
int	"int"
null	"$null$"
string	"$string$"
float	"$float$"
char	"$char$"
const	"$const$"
func	"$func$"
for	"for"
if	"if"
while	"$while$"
true	"$true$"
false	"$false$"
return	"$return$"
else	"$else$"
elif	"$elif$"
class	"$class$"
+	" + "
-	" − "
%	" %"
*	"*"
&	"&"
|	"|"
/	"/"
>	">"
<	"<"
(	"("
)	")"
{	"{"
}	"}"
[	"["
]	"]"
;	";"
,	","
.	"."

Componente Léxico	Expresión Regular
=	" = "
++	" + +"
−−	" − −"
<=	"<="
>=	">="
!=	"! ="
+=	"+ = "
-=	"− = "
=	" = "
/=	"/ = "
<<	"<<"
||	"||"
&&	"&&"
>>	">>"

7. Rúbricas

Rúbrica	Cumple	Cumple con obs.	No cumple
Informe: El informe debe estar en Latex, con un formato limpio (buena presentación) y facil de leer.	5	2.5	0
Implementación: Responde cada ejercicio correctamente.	15	7.5	0
Errores ortográficos: Por cada error ortográfico, se le descontará un punto.	-	-	-