Casos de uso léxico

1. La declaración de identificadores debe comenzar con una letra y posteriormente una combinación de letras y números o solo letras. Si un identificador comienza con un número el analizador léxico denota este error con el error 150.
Ejemplo.

```
ent x6;    <-- correcto
cad abc5;  <-- correcto
ent 45y;   <-- incorrecto
cad 6def;  <-- incorrecto
```

2. Para declarar una variable de tipo cadena se debe comenzar con comillas dobles (“) y cerrar con comillas dobles, si se abre pero no se cierra el analizador léxico marca un token desconocido, si se cierra pero no se abre entonces el analizador lo clasificara como un identificador.
Ejemplo.

```
cad abc;
abc = “hola;   <-- imposible de clasificar
abc = hola”;   <-- identificador
abc = “hola”;  <-- correcto
```

3. Al introducir caracteres fuera de una cadena y que no forma parte del lenguaje el analizador léxico deduce que es un token imposible de clasificar enviando de aviso el mensaje de error 180.
Ejemplo.

```
ent x;
cad abc;
x = 5 % 1;          <--incorrecto
abc = “hola ¬ % ~” <-- aceptable
```

4. El analizador léxico separa en token las palabras reservadas y signos en las funciones *imprime, imprime, si* y *sino,* tanto si hay o no hay espacios entre las funciones y los paréntesis o llaves respectivamente.
Ejemplo.

```
imprime(“hola”);
imprime ( “hola
mientras(x<55){
mientras ( x < 55 ) {          }
```

5. El analizador léxico es capaz de separar en tokens tanto si hay un salto de línea o si hay espacios divisorios entre las palabras, además de separar el punto y coma “;”

como un nuevo token si hay o no espacios entre la expresión y el símbolo. Este caso de uso no aplica al momento de asignar números negativos (véase caso de uso 15). Supongamos el siguiente ejemplo donde se omite por error la llave de la palabra reservada *inicio*.

```
inicio ent x; . . .

inicio
ent x;
. . .
```

la división de tokens es igual, siendo el resultado en la tabla de símbolos los token de la imagen derecha y no los de la imagen izquierda:

```
inicio
ent
x
;
```

```
inicioent
x
;
```

6. El analizador léxico es capaz de identificar a qué tipo corresponde cada token por medio de autómatas que verifican una estructura y así asignan este tipo de manera abreviada a la tabla de símbolos.

Ejemplo.

```
cad abc;
abc = "hola";
```

Lexema	Tipo
cad	Palabra reservada
abc	Identificador
;	Signo delimitador
abc	Identificador
=	Equivalente
“hola”	Signo cadena
;	Signo delimitador

7. El analizador léxico es capaz de revisar en toda la tabla de símbolos buscando el valor de un identificador, si es que tiene, para tener una referencia del valor antes en momento de compilación, antes de la ejecución.

Lexema	Tipo	Valor
cad	Palabra reservada	
abc	Identificador	
;	Signo delimitador	
abc	Identificador	“hola”

=	Equivalente	
“hola”	Signo cadena	“hola”
;	Signo delimitador	

```
cad abc;
abc = “hola”;
```

8. Durante el análisis léxico es posible identificar la dirección de un identificador. Por ejemplo se declara una variable y se hace uso posterior de la misma unas cuantas líneas después, se busca la dirección en la que ya fue declarada para evitar errores en los análisis sintáctico y semántico que son posteriores. La dirección es asignada respecto a la ubicación del identificador en la tabla de símbolos.

Lexema	Tipo	Valor	Dirección
cad	Palabra reservada		
abc	Identificador		18
;	Signo delimitador		
abc	Identificador	“hola”	18
=	Equivalente		
“hola”	Signo cadena	“hola”	
;	Signo delimitador		

```
cad abc;
abc = “hola”;
```

9. A cada token se le asigna un tipo, pero además de esto se le asigna un id numérico, esto para abonar a hacer más simple el análisis sintáctico que se realiza posteriormente ya que el análisis léxico es el encargado de generar una tabla de símbolos bien fundamentada y es más simple comparar valores numéricos que cadenas.

```
cad abc;
abc = “hola”;
```

Lexema	Tipo	Valor	Dirección	ID
cad	Palabra reservada			20
abc	Identificador		18	15
;	Signo delimitador			50
abc	Identificador	“hola”	18	15
=	Equivalente			100
“hola”	Signo cadena	“hola”		80
;	Signo delimitador			50

11. Para declarar un identificador ya sea entero o cadena, el identificador debe iniciar con una letra minúscula, ya que si se inicia con una letra mayúscula marca error, soltándonos el error 150, que un identificador posiblemente fue declarado en caso de un identificador para cadena y el error 180, lexema imposible de calificar para un identificador de entero.
Ejemplo.

```
cad Abc;
ent X;
```

12. El analizador léxico es capaz de contabilizar líneas de código para poder emitir mensajes de error e indicarle al usuario la línea en la que se encuentre un posible error.

13. Al ingresar cualquier carácter que no encaje con la definición del lenguaje como inicio de token, el analizador léxico intenta reconocer el carácter como parte de un patrón (identificador, palabra reservada, etc) y este no encaja en ningún patrón marca error. Marcando el error 130, lexema imposible de calificar.
Ejemplo

```
cad !abc;
ent ¿x;
x = ? + 0;
si [ 0 < 3] { }
ent ñ;
```

14. El analizador léxico es capaz de recuperarse de un error, siendo capaz de crear toda la tabla de símbolos a pesar de que en una línea se encuentre con un error, siendo el analizador léxico capaz de detectar, informar y recuperarse para seguir con el analizador hasta el final.

Ejemplo: el error está en la línea 6, pero aún así el analizador construye toda la tabla de símbolos.

15. Al asignar un número negativo puede haber un máximo de un espacio entre el paréntesis que abre y el signo “-”, de haber más de un espacio se genera una tabla de símbolos equivoca causando errores sintácticos. No debe existir espacios entre el signo menos y el número a negar.
Ejemplo.

```
**correcto**
ent x;
x = (-1);
ent y;
y = ( -5 );
**incorrecto**
ent w;
w = (    -1 );
ent v;
v = (- 1)
```