



# Lenguajes y Autómatas I

## Proyecto Final

**Docente:**  
**Dra. María Italia Jiménez Ochoa**

## Objetivo

Construir un analizador léxico para reconocer las cadenas que pertenecen a un lenguaje regular.

# Metodología



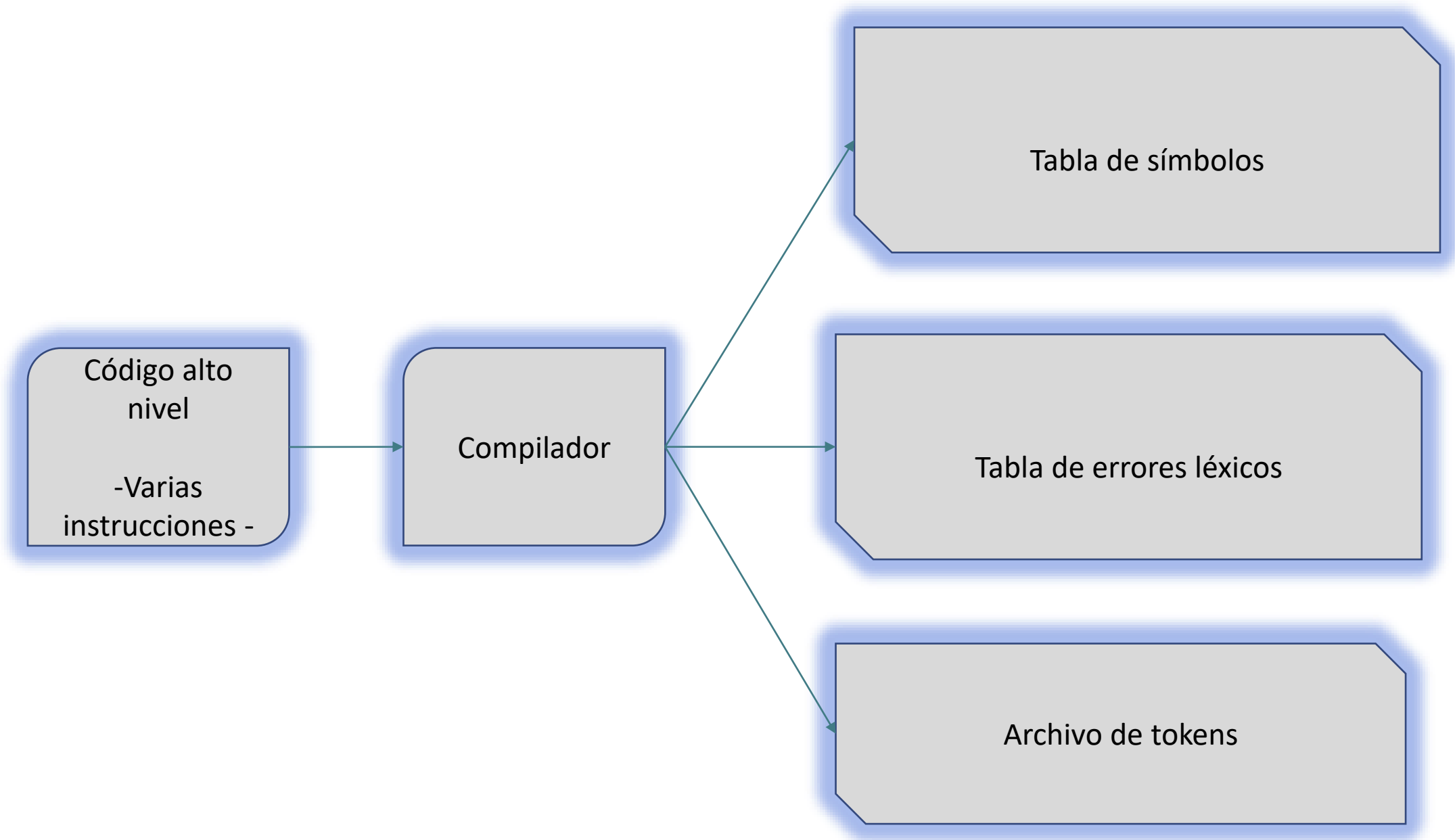
# Metodología

---

1. Este proyecto es continuación del anterior por lo que se realizará en equipo de 2 personas.
2. Investigar los métodos de las clases Pattern y Matcher del package `java.util.regex` de Java.
3. Conocer la sintaxis de Java de las expresiones regulares.
4. Elaborar un programa en Java para implementar las expresiones regulares que representan el patrón de las instrucciones de un lenguaje de programación, utilizando las clases antes mencionadas.

# Compilador





Salidas



## Entrada del programa:

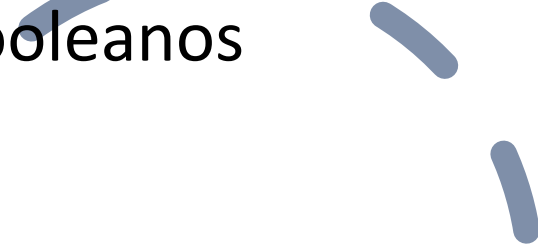
El programa debe permitir ingresar varias instrucciones en la interfaz o desde un archivo de texto.

Las instrucciones pueden contener lexemas (palabras) de tipo:

- Separadores
- Operadores aritméticos
- Operadores relacionales
- Operador de asignación



# Entrada del programa:

- Operadores lógicos o booleanos
  - Identificadores
  - Números enteros
  - Números con punto decimal
  - Tipos de datos
  - Instrucción de acuerdo al número de equipo:
    - If then
    - While
    - For
    - Do while
- 





# Compilador

---

El programa debe verificar que cada lexema cumpla con el patrón del lenguaje de programación (expresión regular).

---

Generar su respectivo **token** e insertarlo a la **tabla de símbolos**. En caso de que no cumpla, generar también un **token de error** e insertarlo a la **tabla de errores**.

# Salida del programa

Tabla de símbolos, con 2 columnas: token y lexema. Presentar la tabla en la interfaz o en el archivo de texto.

Tabla de errores con 4 columnas: token de error, lexema, línea y descripción. Presentar la tabla en la interfaz o en archivo de texto.

Archivo de tokens en un archivo de texto.

# Notas

El token, token de error y lexemas no se deben repetir en la tabla de símbolos y en la tabla de errores.

**El token puede tener cualquier prefijo (ustedes la definen)** o utilizar las sugeridas en la tabla de clasificación de clases anteriores.

En la tabla de errores, en la columna descripción, debe aparecer el texto: error léxico.

# Notas

Pueden estar en cualquier orden los lexemas en la tabla de símbolos pero con su respectivo token.

Implementar en el programa las expresiones regulares que indicaron en el Proyecto 1

En la tabla de errores, no debe aparecer repetido el lexema y el número de renglón.

# Ejemplos



# Ejemplos

---

Supongamos que en mi proyecto 1 indiqué que la expresión regular para

- el identificador tiene la siguiente regla:

Inicia con ITA continua con cualquier secuencia de números y termina con punto.

La expresión regular es: **ITA(0-9)+.**

- Los tipos de datos tienen la siguiente regla:

Comienzan con la palabra Entero, Cadena o Caracter y terminan con –

La expresión regular es: **Entero- U Cadena- U Caracter-**



# Prefijo del token

TIPO DE INSTRUCCIÓN	LENGUAJE REGULAR	EXPRESIÓN REGULAR	PREFIJO DEL TOKEN
Separadores	{(,),{ , ; }	....	SEP
Operadores aritméticos	{+,-,*,/,%}	-----	OA
Operadores relacionales	{<,<=,>,>=,==,! =}	-----	OR
Operador de asignación	{=}	----	OAS
Operadores lógicos o booleanos	{&&,  }	----	OL
Identificadores	<b>Definir la regla o sintaxis. Tiene que ser diferente a la establecida en los lenguajes de programación.</b> Ejemplo. Regla: Inicia con ITA continua con cualquier secuencia de números y termina con punto. Lenguaje regular: {ITA8. , ITA787., ITA111., ....}	<b>ITA(0-9)+.</b>	VAR
Números enteros	<b>Regla. Los números enteros pueden ser positivos o negativos y comienzan con el número del equipo al que estás registrado.</b> {0, 01,-02, 02,....,09,-09, 010,011,...,020,...030,...040,-040,...090,-0100, 0100,,.....}	----	NE



# Prefijo del token

TIPO DE INSTRUCCIÓN	LENGUAJE REGULAR	EXPRESIÓN REGULAR	PREFIJO DEL TOKEN
Números con punto decimal	Los números pueden ser positivos o negativos y la parte entera inicia con el número del equipo al que estás registrado. { -0.01, 0.01, 0.0001, 01.01, ...-067.677., 029093.0009,.....}	.....	NPD
Tipos de datos	Seleccionar al menos 3 diferentes tipos de datos. La sintaxis tiene que ser diferente a la establecida en los lenguajes de programación. Ejemplo {Entero-, Cadena-, Caracter-}	Entero- U Cadena- U Caracter-	TD
Instrucción iterativa o de selección – Equipo 1-	.....	....	....





## Entrada del programa

Entero- ITA5. , ITA67. ;  
ITA87. = ITA123. \* ITA ;

## Salidas del programa

### Tabla de símbolos

Lexema	Token
Entero-	TD1
ITA5.	VAR1
ITA67.	VAR2
ITA87.	VAR3
ITA123.	VAR4
,	SEP1
;	SEP2
=	OAS1
*	OA1

### Tabla de error

Token error	Lexema	Línea	Descripción
ERL1	ITA	2	Error léxico

### Archivo de tokens.txt

```
TD1 VAR1 SEP1 VAR2 SEP2  
VAR3 OAS1 VAR4 OA1 ERL1 SEP2
```



## Entrada del programa

Entero- ITA5. = 08 , ITA67. = 07;  
ITA87. = ITA123. \* ITA000. ;

## Salidas del programa

### Tabla de símbolos

Lexema	Token
Entero-	TD1
ITA5.	VAR1
ITA67.	VAR2
ITA87.	VAR3
ITA123.	VAR4
ITA000.	VAR5
,	SEP1
;	SEP2
=	OAS1
*	OA1
08	NE1
07	NE2

### Tabla de error

Token error	Lexema	Línea	Descripción
-------------	--------	-------	-------------

### Archivo de tokens.txt

```
TD1 VAR1 OAS1 NE1 SEP1 VAR2 OAS1 NE2 SEP2  
VAR3 OAS1 VAR4 OA1 VAR5 SEP2
```



## Entrada del programa

Entero- ITA5. = 08 , ITA67. = 07;  
ITA87. = ITA123. \* ITA000. ;  
ITA67. = ITA5. \* 07 \* 08 ;

## Salidas del programa

### Tabla de símbolos

Lexema	Token
Entero-	TD1
ITA5.	VAR1
ITA67.	VAR2
ITA87.	VAR3
ITA123.	VAR4
ITA000.	VAR5
,	SEP1
;	SEP2
=	OAS1
*	OA1
08	NE1
07	NE2

### Tabla de error

Token error	Lexema	Línea	Descripción
-------------	--------	-------	-------------

### Archivo de tokens.txt

```
TD1 VAR1 OAS1 NE1 SEP1 VAR2 OAS1 NE2 SEP2  
VAR3 OAS1 VAR4 OA1 VAR5 SEP2  
VAR2 OAS1 VAR1 OA1 NE2 OA1 NE1 SEP2
```



## Entrada del programa

Entero- ITA5. = 08 , ITA67. = 07;  
ITA87. = ITA123. \* ITA000. ;  
ITA67. = ITA5. \* 07 \* 99 ;

## Salidas del programa

### Tabla de símbolos

Lexema	Token
Entero-	TD1
ITA5.	VAR1
ITA67.	VAR2
ITA87.	VAR3
ITA123.	VAR4
ITA000.	VAR5
,	SEP1
;	SEP2
=	OAS1
*	OA1
08	NE1
07	NE2

### Tabla de error

Token error	Lexema	Línea	Descripción
ERL1	99	3	Error léxico

### Archivo de tokens.txt

```
TD1 VAR1 OAS1 NE1 SEP1 VAR2 OAS1 NE2 SEP2  
VAR3 OAS1 VAR4 OA1 VAR5 SEP2  
VAR2 OAS1 VAR1 OA1 NE2 OA1 ERL1 SEP2
```



## Entrada del programa

Entero- ITA5. = 08 , ITA67. = 07;  
ITA87. = ITA123. \* ITA000. ;  
ITA67. = ITA5 \* ITA5 \* 99 ;

## Salidas del programa

### Tabla de símbolos

Lexema	Token
Entero-	TD1
ITA5.	VAR1
ITA67.	VAR2
ITA87.	VAR3
ITA123.	VAR4
ITA000.	VAR5
,	SEP1
;	SEP2
=	OAS1
*	OA1
08	NE1
07	NE2

### Tabla de error

Token error	Lexema	Línea	Descripción
ERL1	ITA5	3	Error léxico
ERL2	ITA5	3	Error léxico

**INCORRECTO**

### Archivo de tokens.txt

```
TD1 VAR1 OAS1 NE1 SEP1 VAR2 OAS1 NE2 SEP2  
VAR3 OAS1 VAR4 OA1 VAR5 SEP2  
VAR2 OAS1 ERL1 OA1
```



## Entrada del programa

Entero- ITA5. = 08 , ITA67. = 07;  
ITA87. = ITA123. \* ITA000. ;  
ITA67. = ITA5 \* ITA5 \* 99 ;

## Salidas del programa

### Tabla de símbolos

Lexema	Token
Entero-	TD1
ITA5.	VAR1
ITA67.	VAR2
ITA87.	VAR3
ITA123.	VAR4
ITA000.	VAR5
,	SEP1
;	SEP2
=	OAS1
*	OA1
08	NE1
07	NE2

### Tabla de error

Token error	Lexema	Línea	Descripción
ERL1	ITA5	3	Error léxico
ERL2	99	3	Error léxico

### Archivo de tokens.txt

```
TD1 VAR1 OAS1 NE1 SEP1 VAR2 OAS1 NE2 SEP2  
VAR3 OAS1 VAR4 OA1 VAR5 SEP2  
VAR2 OAS1 ERL1 OA1 ERL1 OA1 ERL2 SEP2
```



## Entrada del programa

Entero- ITA5. = 08 , ITA67. = 07;  
ITA87. = ITA123. \* ITA5 ;  
ITA67. = ITA5 \* ITA5 \* 99 ;

### OPCION NO. 1

## Salidas del programa

### Tabla de símbolos

Lexema	Token
Entero-	TD1
ITA5.	VAR1
ITA67.	VAR2
ITA87.	VAR3
ITA123.	VAR4
,	SEP1
;	SEP2
=	OAS1
*	OA1
08	NE1
07	NE2

### Tabla de error

Token error	Lexema	Línea	Descripción
ERL1	ITA5	2	Error léxico
ERL2	ITA5	3	Error léxico
ERL3	99	3	Error léxico

### Archivo de tokens.txt

```
TD1 VAR1 OAS1 NE1 SEP1 VAR2 OAS1 NE2 SEP2  
VAR3 OAS1 VAR4 OA1 ERL1 SEP2  
VAR2 OAS1 ERL2 OA1 ERL2 OA1 ERL3 SEP2
```



## Entrada del programa

Entero- ITA5. = 08 , ITA67. = 07;  
ITA87. = ITA123. \* ITA5 ;  
ITA67. = ITA5 \* ITA5 \* 99 ;

### OPCION NO. 2

## Salidas del programa

### Tabla de símbolos

Lexema	Token
Entero-	TD1
ITA5.	VAR1
ITA67.	VAR2
ITA87.	VAR3
ITA123.	VAR4
,	SEP1
;	SEP2
=	OAS1
*	OA1
08	NE1
07	NE2

### Tabla de error

Token error	Lexema	Línea	Descripción
ERL1	ITA5	2, 3	Error léxico
ERL2	99	3	Error léxico

### Archivo de tokens.txt

```
TD1 VAR1 OAS1 NE1 SEP1 VAR2 OAS1 NE2 SEP2  
VAR3 OAS1 VAR4 OA1 ERL1 SEP2  
VAR2 OAS1 ERL1 OA1 ERL1 OA1 ERL2 SEP2
```





## Entrada del programa

Entero- ITA5. = 08 , ITA67. = 07;  
ITA87. = ITA123. \* ITA000. ;  
ITA67 = I8TA5. \* 07 \*+ 08 ;

## Salidas del programa

### Tabla de símbolos

Lexema	Token
Entero-	TD1
ITA5.	VAR1
ITA67.	VAR2
ITA87.	VAR3
ITA123.	VAR4
ITA000.	VAR5
,	SEP1
;	SEP2
=	OAS1
*	OA1
08	NE1
07	NE2

### Tabla de error

Token error	Lexema	Línea	Descripción
ERL1	I8TA5.	3	Error léxico
ERL2	*+	3	Erro léxico

### Archivo de tokens.txt

```
TD1 VAR1 OAS1 NE1 SEP1 VAR2 OAS1 NE2 SEP2  
VAR3 OAS1 VAR4 OA1 VAR5 SEP2  
VAR2 OAS1 ERL1 OA1 NE2 ERL2 NE1 SEP2
```



# Entrada del programa

Entero- ITA5. = 08 , ITA67. = 07;  
Cadena- ITA000. ;  
ITA67 = ITA5. \* 07 \* 08 ;

# Salidas del programa

Tabla de símbolos

Lexema	Token
Entero-	TD1
Cadena-	TD2
ITA5.	VAR1
ITA67.	VAR2
ITA000.	VAR3
,	SEP1
;	SEP2
=	OAS1
*	OA1
8	NE1
7	NE2

Tabla de error

Token error	Lexema	Línea	Descripción
-------------	--------	-------	-------------

Archivo de tokens.txt

TD1 VAR1 OAS1 NE1 SEP1 VAR2 OAS1 NE2 SEP2  
TD2 VAR3 SEP2  
VAR2 OAS1 VAR1 OA1 NE2 OA1 NE1 SEP2



# Ejemplo

IF Then



# Prefijo del token

TIPO DE INSTRUCCIÓN	LENGUAJE REGULAR	EXPRESIÓN REGULAR	PREFIJO DEL TOKEN
Separadores	{(,),{ , ; } }	....	SEP
Operadores aritméticos	{+,-,*,/,%}	-----	OA
Operadores relacionales	{<,<=,>,>=,==,! =}	-----	OR
Operador de asignación	{=}	----	OAS
Operadores lógicos o booleanos	{&&,  }	----	OL
Identificadores	<b>Definir la regla o sintaxis. Tiene que ser diferente a la establecida en los lenguajes de programación.</b> Ejemplo. Regla: Inicia con ITA continua con cualquier secuencia de números y termina con punto. Lenguaje regular: {ITA8. , ITA787., ITA111., ....}	<b>ITA(0-9)+.</b>	VAR
Números enteros	<b>Regla. Los números enteros pueden ser positivos o negativos y comienzan con el número del equipo al que estás registrado.</b> {0, 01,-02, 02,....,09,-09, 010,011,...,020,...030,...040,-040,...090,-0100, 0100,,.....}	----	NE



# Prefijo del token

TIPO DE INSTRUCCIÓN	LENGUAJE REGULAR	EXPRESIÓN REGULAR	PREFIJO DEL TOKEN
Números con punto decimal	Los números pueden ser positivos o negativos y la parte entera inicia con el número del equipo al que estás registrado. { -0.01, 0.01, 0.0001, 01.01, ...-067.677.., 029093.0009,.....}	.....	NPD
Tipos de datos	Seleccionar al menos 3 diferentes tipos de datos. La sintaxis tiene que ser diferente a la establecida en los lenguajes de programación. Ejemplo {Entero-, Cadena-, Caracter-}	Entero- U Cadena- U Caracter-	TD
Instrucción iterativa o de selección	{if, then}	If U then	IF



## Entrada del programa

```
Entero- ITA5. = 08 , ITA67. = 07;

if ( ITA000. >= 010 ) then
{
    ITA67. = ITA5. * 07 * 08 ;
}
```

### Tabla de símbolos

Lexema	Token
Entero-	TD1
ITA5.	VAR1
ITA67.	VAR2
ITA000.	VAR3
,	SEP1
;	SEP2
=	OAS1
*	OA1
08	NE1
07	NE2
if	IF1
(	SEP3
>=	OR1
010	NE3
)	SEP4
then	IF2
{	SEP5
}	SEP6



## Salidas del programa

### Tabla de error

Token error	Lexema	Línea	Descripción
-------------	--------	-------	-------------

### Archivo de tokens.txt

```
TD1 VAR1 OAS1 NE1 SEP1 VAR2 OAS1 NE2 SEP2
IF1 SEP3 VAR3 OR1 NE3 SEP4 IF2
SEP5
VAR2 OAS1 VAR1 OA1 NE2 OA1 NE1 SEP2
SEP6
```

## Entrada del programa

```
Entero- ITA5. = 08 , ITA67. = 07;  
if ( ITA000. >= 010 ) THEN  
{  
    ITA67. = ITA5. * 07 * 08 ;  
}
```

### Tabla de símbolos

Lexema	Token
Entero-	TD1
ITA5.	VAR1
ITA67.	VAR2
ITA000.	VAR3
,	SEP1
;	SEP2
=	OAS1
*	OA1
08	NE1
07	NE2
if	IF1
(	SEP3
>=	OR1
010	NE3
)	SEP4
{	SEP5
}	SEP6

## Salidas del programa

### Tabla de error

Token error	Lexema	Línea	Descripción
ERL1	THEN	2	Error léxico

### Archivo de tokens.txt

```
TD1 VAR1 OAS1 NE1 SEP1 VAR2 OAS1 NE2 SEP2  
IF1 SEP3 VAR3 OR1 NE3 SEP4 ERL1  
SEP5  
VAR2 OAS1 VAR1 OA1 NE2 OA1 NE1 SEP2  
SEP6
```



# Ejemplo

While





# Prefijo del token

TIPO DE INSTRUCCIÓN	LENGUAJE REGULAR	EXPRESIÓN REGULAR	PREFIJO DEL TOKEN
Separadores	{(,),{ , ; }	....	SEP
Operadores aritméticos	{+,-,*,/,%}	-----	OA
Operadores relacionales	{<,<=,>,>=,==,! =}	-----	OR
Operador de asignación	{=}	----	OAS
Operadores lógicos o booleanos	{&&,  }	----	OL
Identificadores	<b>Definir la regla o sintaxis. Tiene que ser diferente a la establecida en los lenguajes de programación.</b> Ejemplo. Regla: Inicia con ITA continua con cualquier secuencia de números y termina con punto. Lenguaje regular: {ITA8. , ITA787., ITA111., ....}	<b>ITA(0-9)+.</b>	VAR
Números enteros	<b>Regla. Los números enteros pueden ser positivos o negativos y comienzan con el número del equipo al que estás registrado.</b> {0, 01,-02, 02,....,09,-09, 010,011,...,020,...030,...040,-040,...090,-0100, 0100,,.....}	----	NE



# Prefijo del token

TIPO DE INSTRUCCIÓN	LENGUAJE REGULAR	EXPRESIÓN REGULAR	PREFIJO DEL TOKEN
Números con punto decimal	<b>Los números pueden ser positivos o negativos y la parte entera inicia con el número del equipo al que estás registrado.</b> { -0.01, 0.01, 0.0001, 01.01, ...-067.677.., 029093.0009,.....}	.....	NPD
Tipos de datos	<b>Seleccionar al menos 3 diferentes tipos de datos. La sintaxis tiene que ser diferente a la establecida en los lenguajes de programación.</b> Ejemplo {Entero-, Cadena-, Caracter-}	Entero- U Cadena- U Caracter-	TD
Instrucción iterativa o de selección	{while}	while	WH



# Entrada del programa

```
Entero- ITA5. = 08 , ITA67. = 07;

while( ITA000. >= 010 )
{
    ITA67. = ITA5. * 07 * 08 ;
}
```

Tabla de símbolos

Lexema	Token
Entero-	TD1
ITA5.	VAR1
ITA67.	VAR2
ITA000.	VAR3
,	SEP1
;	SEP2
=	OAS1
*	OA1
08	NE1
07	NE2
while	WH1
(	SEP3
>=	OR1
010	NE3
)	SEP4
{	SEP5
}	SEP6

# Salidas del programa

Tabla de error

Token error	Lexema	Línea	Descripción
-------------	--------	-------	-------------

Archivo de tokens.txt

```
TD1 VAR1 OAS1 NE1 SEP1 VAR2 OAS1 NE2 SEP2
WH1 SEP3 VAR3 OR1 NE3 SEP4
SEP5
VAR2 OAS1 VAR1 OA1 NE2 OA1 NE1 SEP2
SEP6
```

## Entrada del programa

```
Entero- ITA5. = 08 , ITA67. = 07;  
WHILE ( ITA000. >= 010 )  
{  
    ITA67. = ITA5. * 07 * 08 ;  
}
```

### Tabla de símbolos

Lexema	Token
Entero-	TD1
ITA5.	VAR1
ITA67.	VAR2
ITA000.	VAR3
,	SEP1
;	SEP2
=	OAS1
*	OA1
8	NE1
7	NE2
(	SEP3
>=	OR1
10	NE3
)	SEP4
{	SEP5
}	SEP6

## Salidas del programa

### Tabla de error

Token error	Lexema	Línea	Descripción
ERL1	WHILE	2	Error léxico

### Archivo de tokens.txt

```
TD1 VAR1 OAS1 NE1 SEP1 VAR2 OAS1 NE2 SEP2  
ERL1 SEP3 VAR3 OR1 NE3 SEP4  
SEP5  
VAR2 OAS1 VAR1 OA1 NE2 OA1 NE1 SEP2  
SEP6
```



TECNOLÓGICO  
NACIONAL DE MÉXICO



# Ejemplo

Do While



# Prefijo del token

TIPO DE INSTRUCCIÓN	LENGUAJE REGULAR	EXPRESIÓN REGULAR	PREFIJO DEL TOKEN
Separadores	{(,),{ , ; }	....	SEP
Operadores aritméticos	{+,-,*,/,%}	-----	OA
Operadores relacionales	{<,<=,>,>=,==,! =}	-----	OR
Operador de asignación	{=}	----	OAS
Operadores lógicos o booleanos	{&&,  }	----	OL
Identificadores	<b>Definir la regla o sintaxis. Tiene que ser diferente a la establecida en los lenguajes de programación.</b> Ejemplo. Regla: Inicia con ITA continua con cualquier secuencia de números y termina con punto. Lenguaje regular: {ITA8. , ITA787., ITA111., ....}	<b>ITA(0-9)+.</b>	VAR
Números enteros	<b>Regla. Los números enteros pueden ser positivos o negativos y comienzan con el número del equipo al que estás registrado.</b> {0, 01,-02, 02,....,09,-09, 010,011,...,020,...030,...040,-040,...090,-0100, 0100,,.....}	----	NE



# Prefijo del token

TIPO DE INSTRUCCIÓN	LENGUAJE REGULAR	EXPRESIÓN REGULAR	PREFIJO DEL TOKEN
Números con punto decimal	Los números pueden ser positivos o negativos y la parte entera inicia con el número del equipo al que estás registrado. { -0.01, 0.01, 0.0001, 01.01, ...-067.677.., 029093.0009,.....}	.....	NPD
Tipos de datos	Seleccionar al menos 3 diferentes tipos de datos. La sintaxis tiene que ser diferente a la establecida en los lenguajes de programación. Ejemplo {Entero-, Cadena-, Caracter-}	Entero- U Cadena- U Caracter-	TD
Instrucción iterativa o de selección	{do,while}	do U while	DW



# Entrada del programa

```
Entero- ITA5. = 08 , ITA67. = 07;

do
{
    ITA67. = ITA5. * 07 * 08 ;

}
while( ITA000. >= 010 ) ;
```

Tabla de símbolos

Lexema	Token
Entero-	TD1
ITA5.	VAR1
ITA67.	VAR2
ITA000.	VAR3
,	SEP1
;	SEP2
=	OAS1
*	OA1
08	NE1
07	NE2
do	DW1
(	SEP3
>=	OR1
010	NE3
)	SEP4
{	SEP5
}	SEP6
while	DW2



# Salidas del programa

Tabla de error

Token error	Lexema	Línea	Descripción
-------------	--------	-------	-------------

Archivo de tokens.txt

```
TD1 VAR1 OAS1 NE1 SEP1 VAR2 OAS1 NE2 SEP2
DW1
SEP5
VAR2 OAS1 VAR1 OA1 NE2 OA1 NE1 SEP2
SEP6
DW2 SEP3 VAR3 OR1 NE3 SEP4 SEP2
```



## Entrada del programa

```
Entero- ITA5. = 08 , ITA67. = 07;
do
{
    ITA67. = ITA5. * 07 * 08 ;
}
WHILE ( ITA000. >= 010 ) ;
```

### Tabla de símbolos

Lexema	Token
Entero-	TD1
ITA5.	VAR1
ITA67.	VAR2
ITA000.	VAR3
,	SEP1
;	SEP2
=	OAS1
*	OA1
8	NE1
7	NE2
do	DW1
(	SEP3
>=	OR1
10	NE3
)	SEP4
{	SEP5
}	SEP6

## Salidas del programa

### Tabla de error

Token error	Lexema	Línea	Descripción
ERL1	WHILE	6	Error léxico

### Archivo de tokens.txt

```
TD1 VAR1 OAS1 NE1 SEP1 VAR2 OAS1 NE2 SEP2
DW1
SEP5
VAR2 OAS1 VAR1 OA1 NE2 OA1 NE1 SEP2
SEP6
ERL1 SEP3 VAR3 OR1 NE3 SEP4 SEP2
```



# Ejemplo

for



# Prefijo del token

TIPO DE INSTRUCCIÓN	LENGUAJE REGULAR	EXPRESIÓN REGULAR	PREFIJO DEL TOKEN
Separadores	{(,),{ , ; }	....	SEP
Operadores aritméticos	{+,-,*,/,%}	-----	OA
Operadores relacionales	{<,<=,>,>=,==,! =}	-----	OR
Operador de asignación	{=}	----	OAS
Operadores lógicos o booleanos	{&&,  }	----	OL
Identificadores	<b>Definir la regla o sintaxis. Tiene que ser diferente a la establecida en los lenguajes de programación.</b> Ejemplo. Regla: Inicia con ITA continua con cualquier secuencia de números y termina con punto. Lenguaje regular: {ITA8. , ITA787., ITA111., ....}	<b>ITA(0-9)+.</b>	VAR
Números enteros	<b>Regla. Los números enteros pueden ser positivos o negativos y comienzan con el número del equipo al que estás registrado.</b> {0, 01,-02, 02,....,09,-09, 010,011,...,020,...030,...040,-040,...090,-0100, 0100,,.....}	----	NE



# Prefijo del token

TIPO DE INSTRUCCIÓN	LENGUAJE REGULAR	EXPRESIÓN REGULAR	PREFIJO DEL TOKEN
Números con punto decimal	Los números pueden ser positivos o negativos y la parte entera inicia con el número del equipo al que estás registrado. { -0.01, 0.01, 0.0001, 01.01, ...-067.677., 029093.0009,.....}	.....	NPD
Tipos de datos	Seleccionar al menos 3 diferentes tipos de datos. La sintaxis tiene que ser diferente a la establecida en los lenguajes de programación. Ejemplo {Entero-, Cadena-, Caracter-}	Entero- U Cadena- U Caracter-	TD
Instrucción iterativa o de selección	{for}	for	FOR



## Entrada del programa

```
Entero- ITA5. = 08 , ITA67. = 07;

for (ITA0. = 010 ; ITA0. <= 0100 ; ITA0. = ITA0. + 01)
{
    ITA67. = ITA5. * 07 * 08 ;
}
```

### Tabla de símbolos

Lexema	Token
Entero-	TD1
ITA5.	VAR1
ITA67.	VAR2
ITA0.	VAR3
,	SEP1
;	SEP2
=	OAS1
*	OA1
08	NE1
07	NE2
for	FOR1
(	SEP3
<=	OR1
010	NE3
)	SEP4
{	SEP5
}	SEP6
100	NE4
01	NE5
+	OA2



TECNOLÓGICO  
NACIONAL DE MÉXICO



## Salidas del programa

### Tabla de error

Token error	Lexema	Línea	Descripción
-------------	--------	-------	-------------

### Archivo de tokens.txt

```
TD1 VAR1 OAS1 NE1 SEP1 VAR2 OAS1 NE2 SEP2
FOR1 SEP3 VAR3 OAS1 NE3 SEP2 VAR3 OR1 NE4 SEP2
VAR3 OAS1 VAR3 OA2 NE5 SEP4
SEP5
VAR2 OAS1 VAR1 OA1 NE2 OA1 NE1 SEP2
SEP6
```

## Entrada del programa

```
Entero- ITA5. = 08 , ITA67. = 07;  
FOR (ITA0. = 010 ; ITA0. <= 0100 ; ITA0. = ITA0. + 1)  
{  
    ITA67. = ITA5. * 07 * 08 ;  
}
```

### Tabla de símbolos

Lexema	Token
Entero-	TD1
ITA5.	VAR1
ITA67.	VAR2
ITA0.	VAR3
,	SEP1
;	SEP2
=	OAS1
*	OA1
08	NE1
07	NE2
(	SEP3
<=	OR1
010	NE3
)	SEP4
{	SEP5
}	SEP6
0100	NE4
01	NE5
+	OA2



TECNOLÓGICO  
NACIONAL DE MÉXICO



## Salidas del programa

### Tabla de error

Token error	Lexema	Línea	Descripción
ERL1	FOR	2	Error léxico

### Archivo de tokens.txt

```
TD1 VAR1 OAS1 NE1 SEP1 VAR2 OAS1 NE2 SEP2  
ERL1 SEP3 VAR3 OAS1 NE3 SEP2 VAR3 OR1 NE4 SEP2  
VAR3 OAS1 VAR3 OA2 NE5 SEP4  
SEP5  
VAR2 OAS1 VAR1 OA1 NE2 OA1 NE1 SEP2  
SEP6
```



# Observaciones



# Observaciones

En este documento se presentan algunos ejemplos pero en el programa puede haber más lexemas (correctos o incorrectos) como entradas al programa que deben ser analizados.

```
Entero- ITA5. = 8 , ITA67. = 7;  
for (ITA0. = 10 ; ITA0. <= 100 ; ITA0. = ITA0. + 1)  
{  
    ITA67. = ITA5. * 7 * 8 ;  
}  
for (ITA4. = 10 ; ITA0 <= 100 ; ITA0 = ITA0 + 1)  
{  
    ITA67 =+ ITA5. / b ;  
}
```

```
Entero- ITA5. = 08 , ITA67. = 7;  
do  
{  
    ITA67. = ITA5. * 7 * 8 ;  
}  
WHILE ( ITA000. >= 010 ) ;  
W = 23 * 98.6;  
do  
{  
    ITA. = ITA5. / 89.7 ;  
}  
WHILE ( ITA >= 10 && 67 == ITA000.) ;
```





# Observaciones

```
Ent- ITA5. = 8 , ITA67. = 7;  
while( ITA000. >= 10 )  
{  
    ITA67. = ITA5. * 7 * 8 ;  
}  
J = 78.6 + 89.34 * 8;  
while( ITA000. >= 010 && J != 9 )  
{  
    J = 88 + ITA000.  
}
```

```
if ( ITA000. >= 10 ) then  
{    ITA67. = ITA5. * 7 * 8 ;  
}  
  
If ( w != 8 && ITA1. <= 898) then  
{  
    OTRO = INCORRECTO + NADA;  
}  
If (( ITA5.7 == 8 ) then  
{  
    a = 8;  
}
```

Fecha de entrega: 8 de junio de 2021



TECNOLÓGICO  
NACIONAL DE MÉXICO



Entregables  
y  
Fecha de  
entrega

## 1.- PROGRAMA

NOMBRE DEL ARCHIVO:

**6SB\_ANALISISLEXICO\_NOMBREINTEGRANTESDELEQUIPO.RAR**

## 2.- DOCUMENTO:

ALFABETO

EXPRESIÓN REGULAR

DIAGRAMA DE TRANSICION

AUTÓMATA FINITO

**-TABLA CON LA COLUMNA DEL PREFIJO DE TOKENS-**

Fecha de entrega: 8 de junio de 2021



TECNOLÓGICO  
NACIONAL DE MÉXICO



Entregables  
y  
Fecha de  
entrega

REALIZAR LA GRAMÁTICA LIBRE DEL CONTEXTO -GLC- PARA REPRESENTAR LA EXPRESIÓN REGULAR DE

++ LOS IDENTIFICADORES

++ LOS TIPOS DE DATOS.

PARA CADA GLC, OBTENER UN ÁRBOL DE DERIVACIÓN PARA UNA PALABRA CORRECTA Y OTRA INCORRECTA.

NOMBRE DEL ARCHIVO: **6SB\_AL\_DOCTO\_NOMBREINTEGRANTESDELEQUIPO.\***



# Prefijo del token

TIPO DE INSTRUCCIÓN	LENGUAJE REGULAR	EXPRESIÓN REGULAR	PREFIJO DEL TOKEN
Separadores	{(,),{ , ; }	....	SEP
Operadores aritméticos	{+,-,*,/,%}	-----	OA
Operadores relacionales	{<,<=,>,>=,==,! =}	-----	OR
Operador de asignación	{=}	----	OAS
Operadores lógicos o booleanos	{&&,  }	----	OL
Identificadores	<b>Definir la regla o sintaxis. Tiene que ser diferente a la establecida en los lenguajes de programación.</b> Ejemplo. Regla: Inicia con ITA continua con cualquier secuencia de números y termina con punto. Lenguaje regular: {ITA8. , ITA787., ITA111., ....}	<b>ITA(0-9)+.</b>	VAR
Números enteros	<b>Regla. Los números enteros pueden ser positivos o negativos y comienzan con el número del equipo al que estás registrado.</b> {0, 01,-02, 02,....,09,-09, 010,011,...,020,...030,...040,-040,...090,-0100, 0100,,.....}	----	NE



# Prefijo del token

TIPO DE INSTRUCCIÓN	LENGUAJE REGULAR	EXPRESIÓN REGULAR	PREFIJO DEL TOKEN
Números con punto decimal	<b>Los números pueden ser positivos o negativos y la parte entera inicia con el número del equipo al que estás registrado.</b> { -0.01, 0.01, 0.0001, 01.01, ...-067.677.., 029093.0009,.....}	.....	NPD
Tipos de datos	<b>Seleccionar al menos 3 diferentes tipos de datos. La sintaxis tiene que ser diferente a la establecida en los lenguajes de programación.</b> Ejemplo {Entero-, Cadena-, Caracter-}	Entero- U Cadena- U Caracter-	TD
Instrucción iterativa o de selección –por equipo-			