



Universidad Nacional Autónoma de México  
Facultad de Ciencias  
COMPILADORES  
PRÁCTICA 2  
Laura Itzel Rodríguez Dimayuga  
(nombre 2)  
(nombre 3)



### Ejercicio 1

Indica los valores asignados a  $w$ ,  $x$ ,  $y$  y  $z$ . en los siguientes dos códigos estructurados por bloques. Muestrala tabla de símbolos en cada bloque con una implementación imperativa en cada caso:

```
1 int w,x,y,z;  
2 int i = 4; int j = 5;  
3 {  
4     int j =7;  
5     i =6;  
6     w = i+j;  
7 }  
8 x = i+j;  
9 {  
10    int i = 8;  
11    y = i+j;  
12 }  
13 z = i+j;
```

```
1 int w,x,y,z;  
2 int i = 3; int j = 4;  
3 {  
4     int i = 5;  
5     w = i+j;  
6 }  
7 x = i+j;  
8 {  
9     int j = 6;  
10    i = 7;  
11    y = i+j;  
12 }  
13 z = i+j;
```

### Solución.

Variable	Scope	Comentario
int i =4	Bloque 1	Nueva variable
int j = 5	Bloque 1	Nueva variable
int j = 7	Bloque 2	Nueva variable, solo queda en el Bloque 2
i=6	Bloque 2	Afecta al Bloque 1
$x = 6 + 5 = 11$	Bloque 1	Afecta al Bloque 1
int i =8	Bloque 3	Nueva variable, solo queda en el Bloque 3
y = 13	Bloque 3	Afecta al Bloque 1
$z = 6 + 5$	Bloque 1	El valor de i cambio, pero el de j es el del primer region
Final		$w = 13, x = 11, y = 13, z = 11$

Cuadro 1: Tabla de símbolos y valores para el primer código estructurado por bloques

Variable	Scope	Comentario
int i =3	Bloque 1	Nueva variable
int j =4	Bloque 1	Nueva variable
int i = 5	Bloque 1	Afecta solo al Bloque 1
$w = 5 + 4$	Bloque 2	Asignación, afecta al bloque 1
$x = 3 + 4$	Bloque 1	Asignación, afecta al bloque 1
j=6	Bloque 3	Afecta al Bloque 3
i= 7	Bloque 3	Afecta al Bloque 1
int i =8	Bloque 3	Nueva variable, solo queda en el Bloque 3
$y = 7 + 6 = 13$	Bloque 3	Afecta al Bloque 1
$z = 7 + 4 = 11$	Bloque 1	El valor de i cambio, pero el de j es el del primer región
Final		$w = 9, x = 7, y = 13, z = 11$

Cuadro 2: Tabla de símbolos y valores para el segundo código estructurado por bloques

## Ejercicio 2

Divide el siguiente programa en C++ en lexemas y genera los tokens correspondientes:

```
1 float limitedSquare(x) float x; {  
2     /* return x-squared, bit never mover than 100 */  
3     return(x <= -10.0 || x>=10.0) ? 100 : x*x;  
4 }
```

### Solución.

Después del escaneo no tenemos comentarios. Ni espacios en blanco. Lexemas: float, limitedSquare, (, x, ), float, x, ;,, return, (, x, <=, -10.0, ||, x, >=, 10.0, ), ?, 100, :, x, \*, x, ;,

Tokens:

```
1 <float>  
2 <id, limitedSquare>  
3 <(>  
4 <id, x>  
5 <)>  
6 <float>  
7 <id, x>  
8 <;>  
9 <{>  
10 <return>  
11 <(>  
12 <id, x>  
13 <<=>  
14 <float, -10.0>  
15 <||>  
16 <id, x>  
17 <>=>  
18 <float, 10.0>  
19 <)>  
20 <?>  
21 <100>  
22 <:>  
23 <id, x>  
24 <*>  
25 <id, x>  
26 <;>  
27 <}>
```

■

## Ejercicio 3

Define una función recursiva que compute los prefijos de una expresión regular. La base de tal función recursiva es:

$$\begin{aligned} \text{prefix}(\varepsilon) &= \{\varepsilon\} \\ \text{prefix}(a) &= \{a\} \end{aligned}$$

Completa la definición.

**Solución.** Para definir los prefijos de una expresión regular, podemos considerar las siguientes reglas:

Podemos definir las reglas que aplicamos para cada una de las operaciones básicas de las expresiones regulares, si tenemos que  $R$  y  $S$  son expresiones regulares y  $a$  es un símbolo, entonces:

$$\begin{aligned} \text{prefix}(\epsilon) &= \{\epsilon\} \\ \text{prefix}(a) &= \{\epsilon, a\} \\ \text{prefix}(RS) &= \text{prefix}(R) \cup R \circ \text{prefix}(S) \\ \text{prefix}(R|S) &= \text{prefix}(R) \mid \text{prefix}(S) \\ \text{prefix}(R^*) &= R^* \circ \text{prefix}(R) \end{aligned}$$

■

#### Ejercicio 4

Para las siguientes expresiones regulares, da el lenguaje que definen:

1.  $[ab][cd\epsilon]$
2.  $[a - zA - Z]^*at^*$
3.  $ca[tr]$

**Solución.**

1.  $[ab][cd\epsilon]$  define el lenguaje  $\{a, b, ac, ad, bc, bd\}$
2.  $[a - zA - Z]^*at^*$  define el lenguaje de (letras) $at^*$ , es decir, todas cadenas de letras seguidas de **a** y luego de cero o más **t**. Por ejemplo: **at**, **bat**, **cat**, **a**, **ratttt**, etc.
3.  $ca[tr]$  define el lenguaje  $\{cat, car\}$

■

#### Ejercicio 5

Para el siguiente automata, finito no determinista, construye el automata finito determinista equivalente.

**Solución.** 1. Primero, identificamos los estados iniciales:

$$\begin{aligned} S_0 &= \{q_0\} \cup \{r : q \in S_0 \wedge S(q, \epsilon)\} & \text{No hay epsilon transiciones} \\ S_0 &= \{q_0\} \end{aligned}$$

2. Ahora, construimos la función de transición para el autómata determinista. Para cada estado en  $S_0$ ,

determinamos las transiciones para cada símbolo de entrada.

$$\begin{aligned}
\delta(S_0, a) &= \{q_1\} \text{ cupe - closure}(\{q_1\}) \\
&= \{q_1\} = S_1 \\
\delta(S_0, b) &= \emptyset \\
\delta(S_1, a) &= \{q_2\} \cup \epsilon - \text{closure}(\{q_2\}) \\
&= \{q_2\} = S_2 \\
\delta(S_1, b) &= \{q_0, q_2\} \cup \epsilon - \text{closure}(\{q_0, q_2\}) \\
&= \{q_0, q_2\} = S_3 \\
\delta(S_2, a) &= \{q_3\} \cup \epsilon - \text{closure}(\{q_3\}) \\
&= \{q_3, q_0\} = S_4 \\
\delta(S_2, b) &= \emptyset \\
\delta(S_3, a) &= \{q_1, q_3\} \cup \epsilon - \text{closure}(\{q_1, q_2, q_3\}) \\
&= \{q_0, q_1, q_3\} = S_5 \\
\delta(S_3, b) &= \emptyset \\
\delta(S_4, a) &= \{q_1, q_2\} \cup \epsilon - \text{closure}(\{q_1, q_2\}) \\
&= \{q_1, q_2\} = S_6 \\
\delta(S_4, b) &= \emptyset \\
\delta(S_5, a) &= \{q_1, q_2\} \cup \epsilon - \text{closure}(\{q_1, q_2\}) \\
&= \{q_1, q_2\} = S_6 \\
\delta(S_5, b) &= \{q_0, q_2\} \cup \epsilon - \text{closure}(\{q_0, q_2\}) \\
&= \{q_0, q_2\} = S_3 \\
\delta(S_6, a) &= \{q_2, q_3\} \cup \epsilon - \text{closure}(\{q_2, q_3\}) \\
&= \{q_0, q_2, q_3\} = S_7 \\
\delta(S_6, b) &= \{q_0, q_2\} \cup \epsilon - \text{closure}(\{q_0, q_2\}) \\
&= \{q_0, q_2\} = S_3 \\
\delta(S_7, a) &= \{q_1, q_2, q_3\} \cup \epsilon - \text{closure}(\{q_1, q_2, q_3\}) \\
&= \{q_0, q_1, q_2, q_3\} = S_8 \\
\delta(S_7, b) &= \emptyset \\
\delta(S_8, a) &= \{q_1, q_2, q_3\} \cup \epsilon - \text{closure}(\{q_1, q_2, q_3\}) \\
&= \{q_0, q_1, q_2, q_3\} = S_8 \\
\delta(S_8, b) &= \{q_0, q_2\} \cup \epsilon - \text{closure}(\{q_0, q_2\}) \\
&= \{q_0, q_2\} = S_3
\end{aligned}$$

Como no tenemos nuevos estados terminamos. Ahora construimos los estados Finales

$$F = \{S_0, S_3, S_4, S_7, S_8\}$$

■

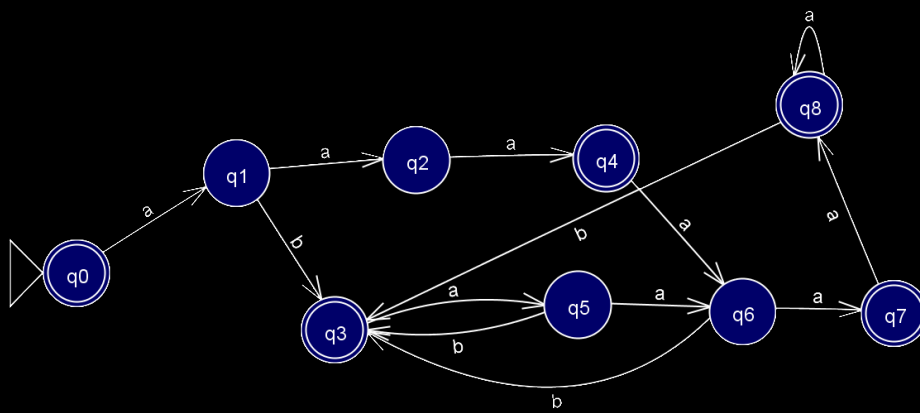


Figura 1: Automata Finito Determinista, ejercicio 5