Slide 1

Análisis léxico

Función:

- Entrada: sucesión de caracteres
- Salida: sucesión de componentes léxicos $< clex, vlex >$
- Creación de la tabla de símbolos
- Eliminación de comentarios
- Relacionar mensajes de error con el programa fuente
- Procesar macros

Separación de tareas:

- Diseño sencillo de cada tarea
- Trabajo especializado: mejora de eficiencia
- Trasportabilidad

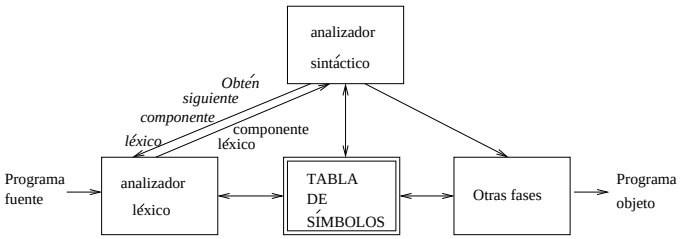
Slide 2

Componentes léxicos

c. lék	lexemas	patrón	val. lék.
eb			
if	If	$(i\|I)(f\|F)$	-
oprel	<	<	MEN
	<=	<=	MEI
	>	>	MAY
	>=	>=	MAI
	=	=	IGU
	<>	<>	DIF
id	x1, inicial	$letra(letra\|digito)^*$	pos. TS
núm	3.141592, 0	$digitos(\varepsilon\|.digitos)\ldots$	su valor

Slide 3

Traducción dirigida por la sintaxis



Slide 4

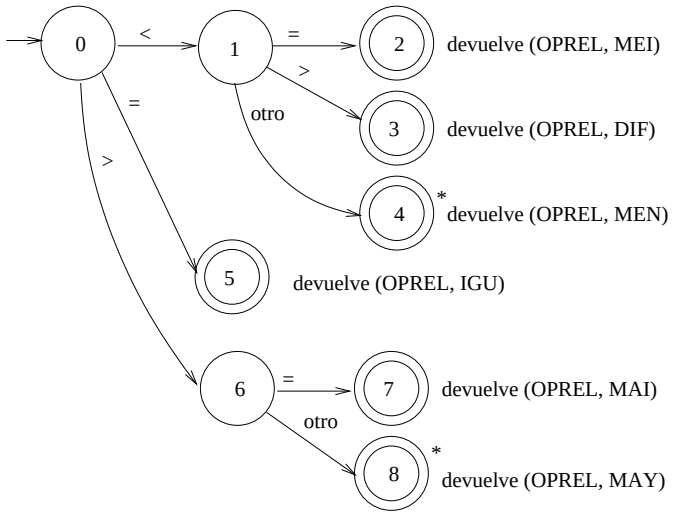
Definiciones regulares

$$
\begin{array}{lcl}
\textbf{letra} & \rightarrow & A|B|\ldots|Z|a|b|\ldots|z \\
\textbf{dígito} & \rightarrow & 0|1|\ldots|9 \\
\textbf{id} & \rightarrow & \textbf{letra}(\,\textbf{letra}\,|\,\textbf{dígito}\,)^{*} \\
\textbf{dígitos} & \rightarrow & \textbf{dígito dígito}^{*} \\
\textbf{frac-opt} & \rightarrow & .\,\textbf{dígitos} \mid \varepsilon \\
\textbf{exp-opt} & \rightarrow & (E(+\mid -\mid \varepsilon)\,\textbf{dígitos}) \mid \varepsilon \\
\textbf{núm} & \rightarrow & \textbf{dígitos frac-opt exp-opt}
\end{array}
$$

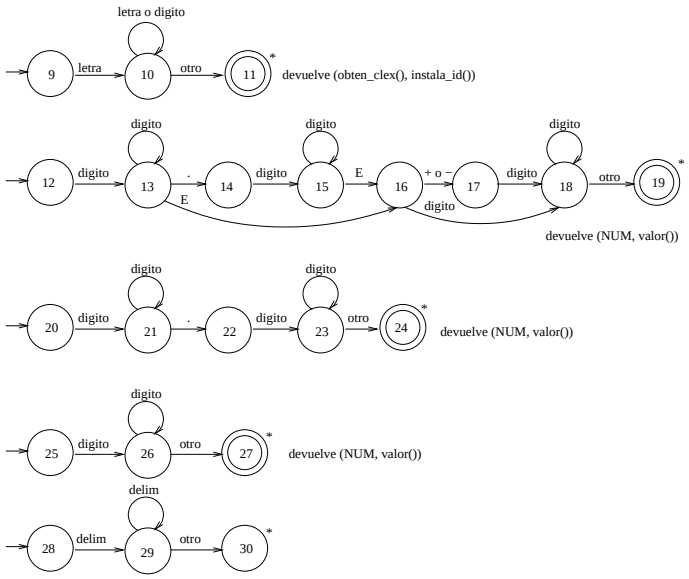
Slide 5

```
leer (c)
case c of
     '=' : devuelve (OPREL, IGU)
     '>' : leer (c)
           case c of '=' : devuelve (OPREL, MEI)
                   other : desleer (c)
                           devuelve (OPREL, MAY)
     '<' : ...
   letra : repeat
               leer (c)
           until c  no sea letra ni dígito
           (c es EOC u otra cosa)
           desleer (c)
           devuelve (ID, lexema hasta aquí) o (IF, ),
                 (según el lexema)
   other : NO SÉ QUÉ ES ESTO
```

Slide 6



Slide 7



Slide 8

```
complex sgte_complex() {
   while (1) {
       switch (estado){
             case 0 : ...  break;
             ...
             case 6 : c = sgtecar();
                      if (c == '=') estado = 7;
                      else          estado = 8;
                      break;
             case 7 : return (OPREL);
                      break;
             case 8 : ... break;
             ...
       }
   }
}
```

Slide 9

```
case 0 : c = sgtecar();
         if (c==blanco || c==tab || c==nueva_linea) {
            estado = 0;
            inicio_lexema++;
         } else if (c == '<') estado = 1;
         else if (c == '=') estado = 5;
         else if (c == '>') estado = 6;
         else estado = fallo();
         break;
case 9 : c = sgtecar();
         if (isletter(c)) estado = 10;
         else estado = fallo();
         break;
case 11: regresa(1); instala_id();
         return (obten_complex());
         break;
```

Slide 10

```
int estado = 0, inicio = 0;
int valor_léxico;

int fallo()
{   delantero = inicio_lexema;
    switch (inicio){
        case 0 :  inicio = 9 ; break;
        case 9 :  inicio = 12; break;
        case 12:  inicio = 20; break;
        case 20:  inicio = 25; break;
        case 25:  recupera();  break;
        default: /* error léxico */
    }
    return inicio;
}
```

Slide 11

```
%{
#define OPREL 257
#define MEN 1
%}
 int yylval;
letra    [A-Za-z]
dig      [0-9]
id       {letra} ( {letra} | {dig} )*

%%
[ \t\n]+  ;
[iI][fF]  return IF;
"<"      { yylval=MEN; return OPREL;}
"<="     { yylval=MEI; return OPREL;}
{id}     { yylval=instala_id(); return ID;}
.        return yytext[0];

%%
int instala_id(){
/* instala el lexema en la T.de Símbolos */
printf ("Instalo %s en T.S.\n", yytext);
}
```