


UNIVERSIDAD NACIONAL DE COLOMBIA
 Departamento de Ingeniería de Sistemas e Industrial
 Felipe Restrepo Calle ferestrepoca@unal.edu.co

Traducción dirigida por la sintaxis (TDS - SDT)

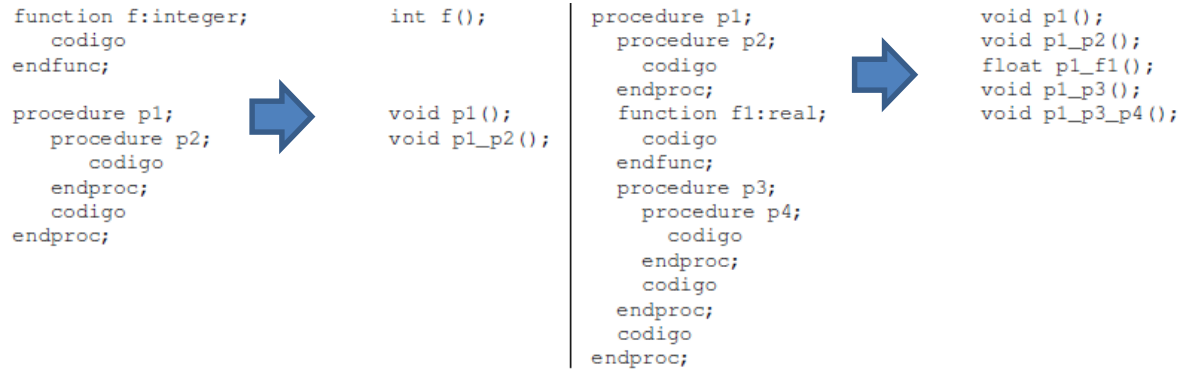
1. Diseñar un Esquema de Traducción Dirigido por la Sintaxis (ETDS) para traducir declaraciones de funciones en C a Pascal. El proceso de traducción se puede especificar con los siguientes ejemplos de traducción:

<code>int f(void),</code>		<code>function f:integer;</code>
<code>g(float a,int *b);</code>		<code>function g(a:real;var b:integer):integer;</code>
<code>void h(int a,float *c),</code>		<code>procedure h(a:integer;var c:real);</code>
<code>j(void);</code>		<code>procedure j;</code>
<code>float f(int a);</code>		<code>function f(a:integer):real;</code>
<code>int f(int a,int b,int c),</code>		<code>function f(a:integer;b:integer;c:integer):integer;</code>
<code>g(int d),</code>		<code>function g(d:integer):integer;</code>
<code>h(int e);</code>		<code>function h(e:integer):integer;</code>

Utilizar como base la siguiente gramática, que genera el lenguaje fuente:

<i>S</i>	→	<i>TipoFun</i> <i>L</i> puntoycoma
<i>TipoFun</i>	→	void
<i>TipoFun</i>	→	int
<i>TipoFun</i>	→	float
<i>L</i>	→	<i>F</i> <i>Lp</i>
<i>Lp</i>	→	coma <i>F</i> <i>Lp</i>
<i>Lp</i>	→	ε
<i>F</i>	→	ident <i>lpar</i> <i>A</i> <i>rpar</i>
<i>A</i>	→	void
<i>A</i>	→	<i>Argu</i> <i>M</i>
<i>M</i>	→	coma <i>Argu</i> <i>M</i>
<i>M</i>	→	ε
<i>Argu</i>	→	<i>Tipo</i> ident
<i>Argu</i>	→	<i>Tipo</i> asterisco ident
<i>Tipo</i>	→	int
<i>Tipo</i>	→	float

2. Diseñar un ETDS para traducir declaraciones de funciones y procedimientos anidados en Pascal a C. El proceso de traducción se puede especificar con los siguientes ejemplos:



Diseñar el ETDS utilizando como base la siguiente gramática, que genera el lenguaje fuente:

$S \rightarrow P$
 $P \rightarrow \text{procedure id ; } L \text{ endproc ;}$
 $P \rightarrow \text{function id : } T \text{ ; } L \text{ endfunc ;}$
 $L \rightarrow P L$
 $L \rightarrow \text{codigo}$
 $T \rightarrow \text{integer}$
 $T \rightarrow \text{real}$

Generación de código intermedio

3. Indicar qué código se genera en CIL para las siguientes instrucciones:

- a. **for** de C/C++, Java, ...

$$Instr \longrightarrow \text{for} (Expr_1 ; Expr_2 ; Expr_3) Instr_1$$

La expresión $Expr_1$ se ejecuta una vez al principio del ciclo; la expresión $Expr_2$ se ejecuta en cada paso del ciclo, y si su resultado es *verdadero* se ejecuta el código de $Instr_1$; la expresión $Expr_3$ se ejecuta después del código de la instrucción en cada paso del ciclo.

- b. **do-while** de C/C++, Java, ...

$$Instr \longrightarrow \text{do } Instr_1 \text{ while } (Expr)$$

La instrucción se ejecuta al menos una vez y se repite mientras la expresión sea *Verdadera*.

- c. **repeat-until** de Pascal

$$Instr \longrightarrow \text{repeat } Instr_1 \text{ until } (Expr)$$

La instrucción se ejecuta al menos una vez y se repite hasta que la expresión sea *Verdadera*.

4. Considerar el siguiente fragmento de la especificación sintáctica de un lenguaje:

$$\begin{array}{ll} S & \longrightarrow L \\ L & \longrightarrow L \mid \\ L & \longrightarrow I \\ I & \longrightarrow \text{break} \\ I & \longrightarrow \text{print entero} \\ I & \longrightarrow \text{switch id llavei } T \text{ llaved} \\ T & \longrightarrow C \mid T \\ T & \longrightarrow D \\ C & \longrightarrow \text{case entero dosp } P \\ D & \longrightarrow \text{default dosp } P \\ P & \longrightarrow L \\ P & \longrightarrow \epsilon \end{array}$$

Diseñar un esquema de traducción dirigida por la sintaxis (ETDS) que genere el código adecuado para CIL y que emita los mensajes de error semántico oportunos.

Consideraciones:

- El único tipo de dato simple de este lenguaje es el tipo entero.
- Esta parte de la gramática caracteriza principalmente la instrucción de selección, que funciona a la manera de C/C++. Considerar, por ejemplo, un programa como el siguiente:

```
switch a {
    case 1:
    case 5:      print 100
    case 2:      print 200
                break
    case 3:      print 300
                break
    default:
}
print 901
```

Si la variable *a* vale 1 o 5 se imprime 100, 200 y 901; si vale 2, se imprime 200 y 901 (la instrucción *break* hace que la ejecución siga tras el *switch*); si la variable *a* vale 3, se imprimirá 300 y 901; finalmente, si tiene cualquier otro valor, se imprime 901

- Asumir que la tabla de símbolos actual es accesible a través de la referencia global *TSA*
- La instrucción *break* no puede aparecer fuera de un *switch*; si lo hace, habrá que emitir el mensaje de error
- Asumir que:
 - los identificadores son siempre variables locales
 - el ETDS obtiene en un atributo sintetizado del símbolo inicial *S* el tamaño mínimo de la pila (necesario para la directiva *maxstack*) para que el código generado se ejecute correctamente en la máquina virtual
 - la signature de la función para imprimir un entero es:
`void [mscorlib]System.Console::Write(int32)`