Lenguaje Micro

- El único tipo de dato es entero.
- Todos los identificadores son declarados implícitamente y con una longitud máxima de 32 caracteres.
- Los identificadores deben comenzar con una letra y están compuestos de letras y dígitos.
- Las constantes son secuencias de dígitos (números enteros).
- Hay dos tipos de sentencias:
  - Asignación
    - `ID := Expresión;`
      Expresión es infija y se construye con identificadores, constantes y los operadores + y –; los paréntesis están permitidos.
  - Entrada/Salida
    - `leer (lista de IDs);`
    - `escribir (lista de Expresiones);`
- Cada sentencia termina con un "punto y coma" (**;**). El cuerpo de un programa está delimitado por **inicio** y **fin**.
- **inicio**, **fin**, **leer** y **escribir** son palabras reservadas y deben escribirse en minúscula.

Gramática Léxica

<token> → *uno de* <identificador> <constante> <palabraReservada> <operadorAditivo> <asignación> <carácterPuntuación>

<identificador> → <letra> {<letra o dígito>}

<constante> → <dígito> {<dígito>}

<letra o dígito> → uno de <letra> <dígito>

<letra> → *una de* **a-z A-Z**

<dígito> → *uno de* **0-9**

<palabraReservada> → *una de* **inicio fin leer escribir**

<operadorAditivo> → *uno de* **+ -**

<asignación> → **:=**

<carácterPuntuación> → *uno de* **( ) , ;**

Tabla de Símbolos

- Estructura de datos compleja que sirve durante toda la fase de análisis para guardar y procesar información.
  - Aporta el contexto, elimina la necesidad de un algoritmo para GSC.
- El diseño depende de cada implementación
- Básicamente contiene cada identificador (como clave) y sus atributos.
- Los atributos dependerán de que tipo de identificador sea, por ejemplo (lista no exhaustiva)
  - Variable: Tipo, ámbito, Lugar de almacenamiento, etc.
  - Función: Tipo, cantidad de parámetros (para cada parámetro el tipo de pasaje)
  - Tipo de dato: Por ejemplo estructuras, arreglos, alias de tipos de datos
  - Palabras reservadas (si el escáner es elemental)
  - Literales cadena.

Tabla de Símbolos Implementación

- En el fondo son una base de tipo clave – valor, se puede implementar como
  - Arreglos
  - Listas encadenada
  - Tablas de Hashing.
- Auxiliares
  - Buffers
    - Todos los literales cadenas seguidos en un buffer y se guardan punteros a ellos.
  - Pilas de TS
    - En lugar de un solo diccionario, uno por cada ámbito y se van apilando. En el caso de un único diccionario, el ámbito forma parte de la “clave”

Tabla de Símbolos - Ejemplos

- Ejemplo función int fx(int x, double y)
  - (fx, funcion, int, 2, int, double)
- Ejemplo int vdatos[20][40];
  - (vdatos, arreglo, int, 2, 20, 40)
- Caso typedef

```
typedef double velocidad;
```

  - Inicialmente el escáner ingresa en la TS a velocidad como un identificador
  - Debe cambiar el atributo identificador a algo que indique “tipo de dato” y su valor sea double
  - La modificación la realiza una rutina semántica
- Devoluciones (caso constante numérica)
  - Token, índice/puntero
  - Token, valor

Centinelas

- En centinela es un carácter “espurio” que indica el final del lexema que se está reconociendo. Es decir, es un carácter que **NO** puede formar parte de ese lexema.
  - Por ejemplo, si está reconociendo un identificador y encuentra un espacio en blanco
  - En general: si encuentra un carácter que no es parte de token que se está reconociendo
- Los LR infinitos ~~SIEMPRE~~ necesitan de un centinela.
- Los LR finitos pueden necesitar o no
  - No necesitan si los lexemas del LR no comparten prefijos
  - Si hay prefijos iguales el lexema más corto necesita de centinela: P.ej: + puede seguir como ++ o encontrar una letra que forma parte de un identificador, esa letra sería el centinela
  - Si bien las palabras reservadas son finitas igual necesitan centinela porque pueden ser prefijo de un identificador. P.ej: for es prefijo de forense

Scanner, implementación directa

- Implementaciones directas
  - Un proceso por cada LR
    - La primer parte detecta la categoría léxica, luego reconoce el lexema en particular.
    - Típicamente un switch sobre el primer carácter leído y subsecuentes aperturas por if o switch
  - Tabla de transición por switch
    - Un loop mientras haya caracteres y un switch en base al estado. En cada uno discrimina según el carácter (o clase de carateres) leído.
    - En alguna época se usaban goto para programarlos, en lugar de variable estado y switch, se saltaba directo desde el estado anterior.

Scanner, implementación directa

- Ventajas
  - El código sigue casi “directamente” la definición del AF
  - Fácil de escribir
  - Rápido al ejecutar
- Contras
  - Mucho código similar
  - Aburrido de escribir si no es un ejemplo simple
  - Casi nada es reutilizable para otro Scanner
  - Se pueden escapar diferencias sutiles con respecto a la definición y se hacen difíciles de detectar y corregir

Seudocódigo - directo

```
estado := INICIAL;
while (verdadero)
    c := Leer caracter
    switch(estado)
        case INICIAL:
            switch(c)
                case 'a': //o sea agrupa
                    estado := ID;
                ...
        case ID:
            ...
end while;
// break o return al encontrar un token
```

Scanner, implementación con tabla de transición

- Es el modo actual para reconocer los identificadores de un lenguaje hipotético.
- Puede ser
  - A mano
  - Con una herramienta como lex/flex
- Filas para los estados y columnas para los caracteres o grupo de caracteres
- Al final dependiendo de si acepta (y cuál fue el estado aceptor) o no ejecutará código que maneje la situación

Scanner, implementación con tabla de transición

- Ventajas
  - Más reutilizable
  - Menos trabajo para implementarlo
  - Si lo hicimos mediante herramientas estamos seguros de cumplir con la definición
- Contras
  - Semi mágico (documentar bien la tabla)
  - Algo más lento que la implementación directa

Seudocódigo para TT

```
estado := INICIAL;
while not debo_parar(estado)
    Leer caracter
    estado := T[estado][caracter];
    /* otras acciones como contar líneas
       Armar lexema, etc */
end while;
if aceptor(estado) then
    if centinela(estado) then
        unput(caracter);
    accept(estado); /* retornar token */
else
    error;/* devolver token de error
             o invocar rutina de manejor de
            error */
end if;
```

Scanner, otras consideraciones

- Los espacios en blanco (espacio, tabulador, nueva línea) suelen ser simplemente ignorados (no siempre, por ejemplo: python)
- En los casos que es necesario un centinela, luego de leerlo se lo devuelve al flujo de entrada.
- Los LR infinitos ~~requieren~~ de un centinela. Los finitos pueden requerirlo o no.
- Es común que al detectar el final de archivo se envíe un Token que lo represente.

Tabla de Transición

TT	L	D	+	-	(	)	,	;	:	=	fdt	esp	otro
0-	1	3	5	6	7	8	9	10	11	14	13	0	14
1	1	1	2	2	2	2	2	2	2	2	2	2	2
2+	99	99	99	99	99	99	99	99	99	99	99	99	99
3	4	3	4	4	4	4	4	4	4	4	4	4	4
4+	99	99	99	99	99	99	99	99	99	99	99	99	99
5+	99	99	99	99	99	99	99	99	99	99	99	99	99
6+	99	99	99	99	99	99	99	99	99	99	99	99	99
7+	99	99	99	99	99	99	99	99	99	99	99	99	99
8+	99	99	99	99	99	99	99	99	99	99	99	99	99
9+	99	99	99	99	99	99	99	99	99	99	99	99	99
10+	99	99	99	99	99	99	99	99	99	99	99	99	99
11	14	14	14	14	14	14	14	14	14	12	14	14	14
12+	99	99	99	99	99	99	99	99	99	99	99	99	99
13+	99	99	99	99	99	99	99	99	99	99	99	99	99
14	99	99	99	99	99	99	99	99	99	99	99	99	99

Comentarios de la TT

Estado	Comentarios
0-	Inicial
1	Reconociendo Identificador, va almacenando cadena
2+	Identificador reconocido. Debe hacer ungetc, comprobar si el identificador es Palabra Reservada (PR), si no es, verificar largo (a lo sumo 32) y almacenar en TS
3	Reconociendo Constante (numérica), va almacenando cadena
4+	Constante reconocida. Debe hace ungetc y en la implementación del libro agregarla a la TS
5+ al 10+	Son casos donde el lexema es un único carácter y no necesita centinela. Por tanto simplemente reconocieron cada uno el token que les corresponde
11	Reconociendo asignación
12+	Asignación reconocida
13+	Fdt reconocido (similar en cierto modo a los casos 5 al 10)
14	Error Léxico

Licencia