Parte I
Analizador Léxico-Sintáctico

Para la realización de esta parte del proyecto se cuenta con la experiencia adquirida en la resolución de los ejercicios de los seminarios S1: "*Introducción al* FLEX" y S2: "*Introducción al* BISON". En realidad, esta parte puede considerarse una extensión de los ejercicios propuestos en ambos seminarios.

Para facilitar el trabajo, en (`/asigDSIC/ETSINF/lppl/pry/P1/`), se proporciona el siguiente material auxiliar:

- `Makefile`. Un fichero de ejemplo para realizar correctamente la tarea de compilación, carga y edición de enlaces de las distintas partes del proyecto.
- `header.h` (En el directorio `include`). Un ejemplo de un posible fichero de cabeceras donde situar las definiciones de constantes y variables globales de `MenosC`. Obviamente, este fichero deberá modificarse por los alumnos para adaptarlo al desarrollo de su propio proyecto.
- `principal.c` (En el directorio `src`). Un ejemplo de un posible fichero con un programa principal y un tratamiento de errores simple.
- `Programas de prueba` (En el directorio `tmp`), Un conjunto de programas de prueba [$a\{0,1,2,3,4,5\}.c$] para comprobar el funcionamiento de esta parte del compilador.

2. Especificación Léxica de `MenosC`

Para la implementación del Analizador Léxico (AL) para `MenosC` se usará la herramienta FLEX[2]. Las restricciones léxicas que se definen para `MenosC` son las siguientes:

- Los nombres de variables pueden contener letras (incluyendo "_") y dígitos, y deben comenzar siempre por una letra. `MenosC` distingue entre mayúsculas y minúsculas. La costumbre es que las variables van en minúscula y las constantes en mayúscula
- Las palabras reservadas deben escribirse en minúscula. La lista de palabras reservadas puede deducirse fácilmente de la gramática del lenguaje que se define en la Figura 1.

[2]Su manual puede encontrarse en `/asigDSIC/ETSINF/lppl/doc/`

- En un programa fuente puedan aparecer constantes enteras y reales; por ejemplo: `28  28.  .55  28.55`
- La constante numérica (`cte`) se considera sin signo. El signo + (ó −) debe tratarse como un símbolo léxico independiente.
- Los comentarios deben ir precedidos por la doble barra (`//`) y terminar con el fin de la linea. Los comentarios pueden aparecer en cualquier lugar donde pueda aparecer un espacio en blanco y solo pueden incluir una linea. Los comentarios no se pueden anidar.
- Los delimitadores se componen de blancos, retornos de linea y tabuladores. Los delimitadores deben ignorarse, excepto cuando deban separar identificadores o palabras reservadas.

3. Especificación Sintáctica de `MenosC`

Para la implementación del Analizador Sintáctico (AS) de `MenosC` se usará la herramienta BISON[3]. La especificación sintáctica para `MenosC` se define en la Figura 1. Como se puede observar, un programa `MenosC` se compone de una secuencia de sentencias entre llaves, bien sean declaraciones de variables o instrucciones, en cualquier orden.

En la gramática, los símbolos terminales son: separadores; operadores; palabras reservadas (en negrita en la gramática); el símbolo **cte**, que representa una constante numérica entera sin signo; y el símbolo **id**, que representa un identificador.

[3]Su manual puede encontrarse en `/asigDSIC/ETSINF/lppl/doc/`

programa → { secuenciaSentencias }
secuenciaSentencias → sentencia | secuenciaSentencias sentencia
sentencia → declaracion | instruccion
declaracion → tipoSimple **id** ; | tipoSimple **id** = constante ;
| tipoSimple **id** [**cte**] ; | **struct** { listaCampos } **id** ;
tipoSimple → **int** | **bool**
listaCampos → tipoSimple **id** ; | listaCampos tipoSimple **id** ;
instruccion → { } | { listaInstrucciones } | instruccionEntradaSalida
| instruccionSeleccion | instruccionIteracion | instruccionExpresion
listaInstrucciones → instruccion | listaInstrucciones instruccion
instruccionEntradaSalida → **read** (**id**) ; | **print** (expresion) ;
instruccionSeleccion → **if** (expresion) instruccion **else** instruccion
instruccionIteracion → **while** (expresion) instruccion
instruccionExpresion → expresion ; | ;
expresion → expresionLogica | **id** operadorAsignacion expresion
| **id** [expresion] operadorAsignacion expresion
| **id** . **id** operadorAsignacion expresion
expresionLogica → expresionIgualdad
| expresionLogica operadorLogico expresionIgualdad
expresionIgualdad → expresionRelacional
| expresionIgualdad operadorIgualdad expresionRelacional
expresionRelacional → expresionAditiva
| expresionRelacional operadorRelacional expresionAditiva
expresionAditiva → expresionMultiplicativa
| expresionAditiva operadorAditivo expresionMultiplicativa
expresionMultiplicativa → expresionUnaria
| expresionMultiplicativa operadorMultiplicatico expresionUnaria
expresionUnaria → expresionSufija | operadorUnario expresionUnaria
| operadorIncremento **id**
expresionSufija → (expresion) | **id** operadorIncremento | **id** [expresion]
| **id** | **id** . **id** | constante
constante → **cte** | **true** | **false**
operadorAsignacion → `=` | `+=` | `-=` | `*=` | `/=`
operadorLogico → `&&` | `||`
operadorIgualdad → `==` | `!=`
operadorRelacional → `>` | `<` | `>=` | `<=`
operadorAditivo → `+` | `-`
operadorMultiplicativo → `*` | `/` | `%`
operadorUnario → `+` | `-` | `!`
operadorIncremento → `++` | `--`

Figura 1: Especificación sintáctica del lenguaje `MenosC`.19