Práctica 1
Un analizador léxico para "miniAda"

Objetivos

1) Establecer e identificar cuáles son los tokens fundamentales para un sencillo lenguaje imperativo
2) Diseñar un analizador léxico para dicho lenguaje
3) Manejar la herramienta Flex, un generador de analizadores léxicos

Contenidos

En esta práctica se propone el desarrollo de un analizador léxico para un subconjunto del lenguaje Ada. Este lenguaje, denominado miniAda, será manejado y ampliado durante el resto de las prácticas.

miniAda es un lenguaje cuyas características principales vamos a enumerar. Si bien el lenguaje no es muy rico, sus elementos son suficientes para trabajar con los conceptos fundamentales del diseño de un traductor.

Las características de miniAda son las siguientes:

1. Los tipos escalares predefinidos son: **`character`**, **`integer`** y **`boolean`**
2. El lenguaje permite el uso de constantes de tipo cadena, pero sólo a efectos de escritura. Las constantes de tipo cadena se representan entre comillas dobles: "hola, caracola", "H", etc. Cuando dentro de una cadena se necesite el carácter doble comilla, éste se indicará mediante dos dobles comillas seguidas, de manera que la ejecución de

```
put("Dijo ""No me lo creo""")
```

   imprimiría por stdout

```
Dijo "No me lo creo"
```

3. Las constantes de tipo carácter se delimitan por comillas simples (apóstrofo): `'a'`,`'A'`,`'*'`, etc. El propio carácter apóstrofo se denota como `''''` (es decir, 3 apóstrogos seguidos).
4. El único constructor de tipos es **`ARRAY`**, cuyo rango se establece exclusivamente mediante la construcción "`v1..v2`", siendo v1 y v2 dos constantes enteras no negativas. El rango de índices va desde v1 hasta v2.
5. Se permite el uso de procedimientos y funciones anidadas.

6. No existe el paso de parámetros de tipo función o procedimiento.
7. Los procedimientos de escritura presentes en el lenguaje son `put` y `new_line`, y admiten constantes de tipo cadena, además de expresiones de tipo entero y carácter (ambos actúan sobre la salida estándard). El hecho de considerarlos presentes implica que no es necesario ninguna claúsula "`with ... use ...`" en los fuentes del lenguaje.
8. El procedimiento `get` se supone definido y primitivo del lenguaje, tanto para variables de tipo carácter como entero o booleano.
9. El procedimiento `skip_line` salta los caracteres en la entrada estándar hasta pasar el primer salto de línea.
10. Las estructuras de control permitidas en miniAda son las que se deducen a partir de la batería de ejemplos en el directorio de salidas de la asignatura.
11. El lenguaje no hace distinción entre mayúsculas y minúsculas en identificadores o palabras reservadas, aunque sí en el caso de valores constantes de tipo cadena o carácter.
12. La declaración de variables no permite su inicialización.
13. Los identificadores empiezan por una letra, y pueden ir seguidos de 0 ó más letras, dígitos o "_" (*underscore*), pero no se admiten dos *undescores* seguidos.
14. Una función solo puede devolver un dato, y ha de ser de un tipo escalar (es decir, no puede devolver vectores).
15. Admite las tres formas de paso de parámetros habituales de Ada.
16. Al igual que en Pascal, los comentarios se abren mediante '`{`' ó '`(*`' y se cierran mediante '`}`' ó '`*)`'. Nótese que esta especificación no respeta los habituales comentarios de Ada.
17. Aquellas cuestiones de definición del lenguaje que no queden claras deben consultarse con el profesor.
18. Existe la instrucción vacía, que se denomina `null`.
19. Todo tipo de dato ha de tener un nombre. Así, tanto en la declaración de variables como de parámetros formales, serán incorrectas cosas como las que siguen:

```
v: array(1..9) of integer;

procedure p(w: array(1..9) of integer) is …
```

Lo correcto será

```
type vect9 is array(1..9) of integer;
v: vect9;

procedure p(w: vect9) is …
```

La práctica pide:

1) Definir los tokens de miniAda. Esto se puede llevar a cabo mediante la definición de constantes o bien mediante el uso de un tipo enumerado. Tener presente que, cuando se implemente el traductor completo, estas definiciones las generará Bison.

2) Construir un analizador léxico para el lenguaje. De momento se trata únicamente de realizar el análisis léxico. Cuando el analizador reconozca un token, se deberá escribir por la salida **standard** información que permita identificar el token reconocido, así como su localización (de la manera que se muestra más adelante). El nombre del fichero a analizar, con el fuente, se debe suministrar como un parámetro en la invocación al analizador.

3) Respecto a los errores léxicos, el analizador construído debe:

A) Dar un **AVISO** (warning) o un **FATAL** (error no recuperable) cuando detecte un error léxico, indicando el tipo de error detectado y la fila y columna en que se ha detectado. Un aviso informa por **stderr** sobre el tipo de error encontrado (ver tabla que aparece a continuación) y aplica una estrategia de recuperación de errores, siguiendo con el análisis. Un error **FATAL** supone la salida del analizador, además de informar por **stderr** de la naturaleza del error.

B) Estrategias de recuperación:

- en el caso de encontrar un carácter extraño, saltarlo, interpretando que se debía tratar de un espacio en blanco.
- en el caso de problemas en comentarios ejecutar las recuperaciones propuestas en la siguiente tabla:

situación	mensaje
(*...}	AVISO! (20,15) Símbolos distintos para un mismo comentario. Sustituyendo '}' por '*)' lexema: }
{...*)	AVISO! (20,15) Símbolos distintos para un mismo comentario. Sustituyendo '*)' por '}' lexema: *)

{...{
{...(*
(*...(*
(*...{
	FATAL! (20,15)
	Comentarios anidados
	lexema: ... (el que corresponda)

{...EOF (*...EOF	FATAL! (20,15) Comentario no terminado lexema: <<EOF>>

...} ...*)	FATAL! (20,15) Comentario no iniciado lexema: ... (el que corresponda)
...#...	AVISO! (20,15) Carácter extraño. Sustituyendo por "blanco" lexema: # (en cada caso, el que corresponda)

- en el caso de detectar un salto de línea o un tabulador una vez que se haya abierto una cadena y aún no se haya cerrado, dará el siguiente aviso:

".....	AVISO! (20,15) Cadena no terminada. Inserto fin de cadena. lexema: \t (ó \n, según el caso)

Resultados

Como resultado de esta práctica hay que entregar el fichero denominado 'pract1.tar'. Este fichero se obtendrá a partir del comando 'tar' de UNIX (ejecutar 'man tar' para tener información sobre él), de manera que la ejecución de la instrucción 'tar -xvf pract1.tar' genere un directorio denominado 'pract1' con los siguientes ficheros:

- fichero 'AL_miniAda_1.l' que contiene el fuente Flex del analizador léxico.
- fichero 'AL_miniAdaMake' que contiene el fichero Make para generar el ejecutable del analizador léxico, de manera que la invocación

```
make -f AL_miniAdaMake
```

genere el ejecutable 'AL_miniAda' que realiza lo que se indica a continuación. Supongamos que hay un fichero fuente en miniAda denominado miFuente. La invocación

```
AL_miniAda miFuente
```

debe escribir en salida standard, en distintas líneas, la información relativa a los tokens reconocidos:

```
(1,1):    IDENTIFICADOR
(125,17): op. ASIGNACION
          ..............
```

donde (f,c) indican la fila y la columna del inicio del lexema y lo que sigue es cualquier información que sirva para ver que se ha identificado el token correspondiente.

Notas

- Como paso previo a la corrección de las prácticas se ejecutará de manera automática un script que comprobará si lo entregado por el alumno como resultado de la práctica es exactamente lo que se pide (ejecución de la descompresión mediante 'tar' y comprobación de que se han generado los dos ficheros pedidos). Si esto no es así, la práctica aparecerá como no presentada. En caso de dudas respecto a lo que se tiene que generar, consultar con el profesor.
- Para lo relativo al tratamiento de los errores, así como de comentarios, será de gran ayuda utilizar las "*start conditions*" (ver documentación de Flex).

Cómo entregar el material pedido en las prácticas

El directorio "/users2/COMPI" será utilizado para las prácticas de esta asignatura. Dentro de él hay dos directorios, denominados "entradas" y "salidas".

El directorio entradas

Contiene un directorio por cada alumno matriculado en la asignatura, cuyo nombre coincide con el de su *username* en Merlin. Es allí donde quedarán depositados los ficheros que cada alumno someta al realizar sus prácticas. La forma de someter un fichero se describe más adelante. El estudiante cuyo *username* coincida con el subdirectorio podrá únicamente hacer 'ls' con el fin de comprobar cuál es el contenido de dicho subdirectorio, pero nada más.

El directorio salidas

En el directorio "salidas" iré depositando ficheros necesarios para el desarrollo de las prácticas (información, bibliotecas necesarias, enunciados de las prácticas en formato PostScript o pdf, etc.).

Para entregar los resultados de las prácticas

Existe una utilidad, denominada 'someter', que es la encargada de depositar los ficheros a entregar en el lugar adecuado. La invocación requiere dos parámetros: el primero, el nombre correspondiente a la asignatura ('COMPI' en este caso); el segundo, el nombre del fichero a someter. Un ejemplo de uso sería el siguiente. Supongamos que hay que entregar el fichero 'pract1.tar'. La invocación sería


```
someter COMPI pract1.tar
```

El programa se encarga de depositar una copia del fichero 'pract1.tar' en el directorio de entrega (dentro de "entradas") del alumno que lo invoca (es decir, en el directorio cuyo nombre coincida con el username de la cuenta desde que se invoca). La invocación sin parámetros recuerda cómo ha de ser la invocación correcta. Tener presente que, una vez sometido un fichero, no puede volver a ser "resometido". En caso de necesidad, contactar con el profesor.

Es preciso tener presentes las siguientes consideraciones:

- Los nombres de los ficheros que se entregan deben coincidir **exactamente** con los nombres que se piden en los enunciados de las prácticas (tener presente que UNIX distingue mayúsculas de minúsculas).
- Es aconsejable mirar el contenido de 'salidas' cuando se vaya a trabajar en alguna práctica; es posible que deje información de última hora. Por ejemplo, para la práctica 1 puede haber información adicional en un fichero denominado 'pract1LEEME'.
- Es interesante que para cuestiones que querais comentar con el profesor enviéis un mensaje a la cuenta 'ezpeleta@unizar.es' (aunque, de preferencia, usar la interacción directa en las clases de prácticas y en las tutorias: cuesta menos hablar que escribir).
- Los ficheros que tengan el sufijo "ps" corresponden a un formato especial (Post Script), y deben ser visualizados utilizando una aplicación específica. En Merlin podeis usar el visualizador "ghostview", ejecutando

```
/usr/local/bin/X11/ghostview nombreArchivo&
```

- Los ficheros que tengan el sufijo "pdf" corresponden a un formato especial (PDF), y deben ser visualizados utilizando una aplicación específica. En Merlin podeis usar el visualizador "acroread", ejecutando

```
/opt/acroread/bin/acroread nombreArchivo&
```

Un último comentario

Una vez que una práctica ha sido entregada, y salvo justificadas excepciones, no podrá volver a ser sometida. Esto hace necesario que el material que se someta esté cuidadosamente revisado (programas funcionando correctamente, fuentes adecuadamente comentados, ficheros "Make" con los "paths" correctos, comentarios sin faltas de ortografía, etc.).

Dado que la manera fundamental de comprobar la corrección de un traductor se basa en su ejecución con una batería de tests, es recomendable que, poniéndoos de

acuerdo entre varios alumnos, prepareis una batería de programas fuente en miniAda que sirva para comprobar el correcto funcionamiento de los programas a realizar en las prácticas. Esta batería debería contener programas, tanto correctos como incorrectos, que cubran todas las características del lenguaje. En cualquier caso, en el directorio de salidas podeis encontrar un pequeño banco de pruebas para miniAda. Cualquier duda sobre el lenguaje debe ser preguntada al profesor de la asignatura.

Fecha límite para entrega de resultados:

Obligatoriamente, cada grupo debe entregar la práctica primera al final de su segunda sesión de prácticas. Los alumnos que no acudan a las sesiones de prácticas deberán entregarlas a la vez que el grupo 4-A, que es el primero que empieza.