Universidad de La Laguna

ESIT – INFORMÁTICA
Departamento de Ingeniería Informática y de Sistemas

Computabilidad y Algoritmia
Curso 2019-2020

PRACTICA 3: Calculadora para lenguajes formales

3.1. Objetivos

El objetivo de la práctica es trabajar los conceptos básicos sobre alfabetos, cadenas y lenguajes formales a través del diseño de una calculadora para lenguajes formales.

Antes de comenzar a describir la herramienta a desarrollar es importante tener claramente identificados los tres conceptos introducidos en el primer tema de la asignatura:

- Un **alfabeto** es un conjunto no vacío y finito de símbolos.
- Una **cadena** es una secuencia finita de símbolos de un determinado alfabeto.
- Un **lenguaje** (formal) es un conjunto de cadenas.

Además de repasar estos conceptos y asentar los principios de las operaciones básicas con cadenas y con lenguajes, se propone que el alumnado utilice este ejercicio para continuar poniendo en práctica los aspectos del desarrollo de programas en C++ sobre los que ya se ha trabajado en las prácticas anteriores, enfatizando al menos los siguientes:

- Paradigma de programación orientada a objetos: identifique clases y objetos que permitan modelar adecuadamente el escenario de trabajo que se planeta.
- Paradigma de modularidad: el programa debiera escribirse de modo que las diferentes funcionalidades que se precisen fueran encapsuladas en métodos concretos cuya extensión textual se mantuviera acotada.
- Formato propuesto para la escritura de programas en C++ en esta asignatura.
- El ejercicio plantea nuevamente una oportunidad para trabajar con conjuntos de la STL [2], [3]. Identifique situaciones en las que el uso de conjuntos y la operatoria de los mismos facilite la implementación que se realiza.

Si el alumnado tiene dudas respecto a cualquiera de estos aspectos, debiera acudir al foro de discusiones de la asignatura para plantearlas en el foro. Se espera que a través de ese foro el alumnado intercambie experiencias y conocimientos, ayudándose mutuamente a resolver dichas dudas. También el profesorado de la asignatura intervendrá en las discusiones que pudieran suscitarse, si fuera necesario.

3.2. Rúbrica de evaluación del ejercicio

Señalamos en este apartado los aspectos más relevantes (la lista no es exhaustiva) que el profesorado tendrá en cuenta a la hora de evaluar el trabajo que el alumnado presentará en la sesión de evaluación de la práctica.

Los principales elementos a evaluar son los siguientes:

- Orientación a objetos: el programa ha de seguir ese paradigma de programación.
- Modularidad: el código ha de ser modular.
- El comportamiento del programa debe ajustarse a lo solicitado en el enunciado.
- El código debe estar escrito de acuerdo a los criterios léxicos de escritura de programas en la asignatura (Google C++ Style Guide).
- Capacidad del programador(a) de introducir cambios en el programa desarrollado.

3.3. Ejercicio práctico

Desarrollar un programa cliente que, dado un fichero de texto de entrada y un código de operación, ejecute las operaciones correspondientes sobre los lenguajes definidos en el fichero de entrada. El resultado de las operaciones llevadas a cabo se almacenará en un fichero de salida cuyo nombre también se tomará por línea de comandos.

Teniendo esto en cuenta la ejecución del programa en línea de comandos sería algo como lo siguiente:

```
./program filein.txt fileout.txt codigo
```

Donde el código de operación será un número entero que identificará a las operaciones tal y como se indica a continuación:

1. Concatenación
2. Unión

3. Intersección
4. Diferencia
5. Sublenguajes
6. Igualdad de lenguajes
7. Inversa
8. Potencia
9. Cierre positivo
10. Cierre de Kleene

Hay que tener en cuenta que para las operaciones binarias (concatenación, unión, intersección y diferencia), se necesitarán dos lenguajes de entrada y se obtendrá un lenguaje de salida. Para comprobar si un lenguaje es un sublenguaje de otro o para comprobar si dos lenguajes son iguales también se necesitarán dos lenguajes de entrada, aunque como salida obtendremos un valor de tipo booleano. Para las operaciones de inversa, potencia, cierre positivo y cierre de Kleene, bastará con tener un único lenguaje de entrada. En este último caso el resultado a obtener será un lenguaje, que además, en el caso del cierre positivo y del cierre de Kleene podrá ser infinito.

Los lenguajes que utilizaremos para hacer las operaciones correspondientes vendrán especificados en el fichero de entrada tal y como se muestra a continuación con algunos ejemplos:

```
{abc, bbcc, a, cab}
{a, aa, aaa}
{&}
{0, 1, 00, 01, 10, 11}
```

De esta forma, cada línea del fichero de entrada contiene, entre llaves, las cadenas que conforman un lenguaje. En el primero de los lenguajes de entrada del ejemplo anterior las cuatro cadenas que conforman el lenguaje se especifican entre llaves y se separan por comas. Teniendo esto en cuenta, las llaves, los espacios y las comas no deberían usarse como símbolos de los alfabetos sobre los que se forman los lenguajes.

Además, cabe destacar que para especificar la cadena vacía vamos a utilizar el símbolo &. Por lo tanto, para evitar confusiones, el símbolo & tampoco podrá utilizarse como elemento de los alfabetos que se utilizarán en la calculadora. A modo de ejemplo, la tercera línea del fichero de entrada anterior definiría al lenguaje que contiene únicamente a la cadena vacía.

Por otra parte, para especificar el lenguaje vacío se propone hacer algo como lo siguiente:

```
{}
```

Por último, en lo que respecta a la especificación de los lenguajes, supondremos que en este ejercicio todos los lenguajes de entrada tendrán un número finito de cadenas.

Una vez clarificado cómo será la especificación de lenguajes en el fichero de entrada, procederemos a realizar la operación correspondiente en función del valor introducido como `código` (último de los argumentos que se pasan) en la línea de comandos. Si la operación requiere de un único lenguaje de entrada, se procesará línea a línea del fichero de entrada aplicando la operación correspondiente y escribiendo el resultado (ya sea un lenguaje o un valor de tipo booleano) en el fichero de salida especificado por línea de comandos. Si la operación es binaria (requiere dos lenguajes como operandos), se tomará entonces el primer lenguaje especificado en el fichero de entrada como primer operando y el segundo de los lenguajes como segundo; luego se tomará el tercer y el cuarto lenguaje del fichero de entrada como siguientes operandos y así hasta que no queden más lenguajes sobre los que realizar las operaciones. En este caso, si el número de lenguajes especificados en el fichero de entrada es impar, no se utilizará el último de los lenguajes para operar. Cada uno de los resultados obtenidos de estas operaciones que requieren de dos lenguajes, se escribirá en el fichero de salida.

Los lenguajes obtenidos como resultado de una operación se escribirán entre llaves en el fichero de salida. Cabe destacar que, para aquellas operaciones que como resultado obtengan un lenguaje con infinitas cadenas, se deberá al menos enumerar diez cadenas del lenguaje, seguidas de tres puntos suspensivos para indicar que dicho lenguaje es infinito. Por este motivo, los puntos tampoco podrán utilizarse como símbolos de los alfabetos.

Hay que tener en cuenta que, con las pautas que se han fijado hasta el momento, se están especificando en cada caso las cadenas que componen el lenguaje, pero el usuario no está especificando explícitamente los elementos del alfabeto sobre el que se forman las mismas. Si se quisiera asociar un alfabeto al lenguaje, sería necesario incluir en el mismo todos los símbolos que forman parte de las cadenas. Caso especial sería el del lenguaje vacío y el del lenguaje que contiene únicamente a la cadena vacía. En este sentido, hay que insistir en que un alfabeto es un conjunto no vacío.

Bibliografía

[1] Transparencias del Tema 1 (Alfabetos, cadenas y lenguajes) de la asignatura, https://campusvirtual.ull.es/1920/mod/resource/view.php?id=20983

[2] STL sets, http://www.cplusplus.com/reference/set/set/

[3] STL sets example, https://tinyurl.com/cyasetexample

[4] Google C++ Style Guide, https://google.github.io/styleguide/cppguide.html