Universidad de La Laguna

E. T. S. DE INGENIERÍA INFORMÁTICA
Departamento de Estadística, I.O. y Computación
Teoría de Autómatas y Lenguajes Formales

PRACTICA 4: Implementación de conjuntos en C++

4.1. El Problema

El lenguaje C++ no tiene (al contrario que Pascal) un tipo básico para representar conjuntos. En otras prácticas de TALF utilizaremos profusamente conjuntos. El objetivo de esta práctica es implementar una clase que permita operaciones básicas con conjuntos.

4.2. Descripción de la práctica

Se definirá una clase `Set` con los atributos y métodos convenientes para representar conjuntos.

Para representar internamente los conjuntos se puede utilizar cualquier idea, no obstante sugerimos aquí una que puede dar buenos resultados. Para representar un conjunto de enteros utilizaremos los bits de un valor de tipo `long`. Si el bit i-ésimo está a 1 ello indicará que el número *i* pertenece al conjunto. Si ese bit está a 0, esto indica que el número *i* no pertenece al conjunto. De este modo podemos representar conjuntos con tantos números naturales como bits tiene un valor de tipo `long` (esto es, `sizeof(long)`). Si deseamos representar conjuntos con un número mayor de elementos, basta usar más de un valor de tipo `long` (es decir, con un vector de `M` valores de tipo `long` podremos representar conjuntos con un máximo de `8* M * sizeof(long)` elementos. El número de valores `long` que se utilizan en la representación interna del conjunto debería ser uno de los atributos de la clase que se ha de definir. Utilizando aritmética de bits es muy fácil implementar las diferentes operaciones sobre conjuntos.

Se definirán al menos dos constructores. El primero de ellos no tiene argumentos y definirá un conjunto que tendrá como máximo `sizeof(long)` elementos. El segundo constructor tiene un parámetro que indica el número máximo de elementos que podremos almacenar en el conjunto. De este modo podremos definir:

```
Set set1, set2(100);
```

El conjunto `set1` podrá almacenar un máximo de `sizeof(long)` elementos, mientras que `set2` puede almacenar hasta 100 elementos.

Al menos se implementarán las siguientes operaciones sobre conjuntos:

1. Unión (+)

2. Complemento relativo (−)

3. Intersección (∗)

4. Complementación (!)

5. Asignación (=)

A la hora de calcular el complementario de un conjunto, se considerará que el conjunto universal es aquél que tiene el máximo número de elementos posible en función de su representación interna.

Se sobrecargará también el operador == de modo que permita determinar si dos conjuntos son iguales. Se definirán métodos que permitan las siguientes operaciones:

1. insertar un elemento en un conjunto.

2. eliminar un elemento de un conjunto (supuesto que ya pertenece al mismo).

3. Vaciar un conjunto (eliminar todos los elementos que lo componen).

4. Determinar si un conjunto está vacío.

5. Determinar si un determinado elemento pertenece al conjunto.

Se sobrecargarán convenientemente los operadores de inserción y extracción en flujos de forma que sea posible leer y escribir un conjunto. A efectos de entrada/salida el conjunto $A = \{1, 2, 3\}$ se representará como $\{1, 2, 3\}$, es decir, listando sus elementos separados por comas y englobados entre llaves.

Para comprobar el correcto funcionamiento de la práctica se implementará una calculadora de conjuntos: desde un fichero (cuyo nombre se suministrará en línea de comandos) se extraerán lineas que contienen expresiones cuyos operandos son conjuntos. El programa escribirá en otro fichero (cuyo nombre es el segundo parámetro que se pasará a la línea de comandos) el resultado de cada una de las expresiones del primer fichero.