Trabajo práctico 3: Implementación
Enrutador de tráfico IP

Normativa

Fechas de entrega: Lunes 14, Miércoles 16, Viernes 18 y Miércoles 23 de diciembre. Todos los días de 17 a 20 horas, excepto el Miércoles 16, que es de 17 a 22 horas.

Normas de entrega: Las contenidas en la página web de la materia.

Enunciado

El TP consta de implementar en C++ lo diseñado en el TP2. Cada grupo trabaja sobre su propio diseño. Se puede usar cualquier compilador y entorno, pero la corrección se realizará exclusivamente compilando en la consola (como fue visto en clase) y usando el compilador gcc instalado en el laboratorio 5.

Se deben documentar y justificar las diferencias no triviales entre el diseño entregado en el TP2 y lo implementado. Se aceptarán modificaciones que mejoren el diseño y también realizadas por cuestiones técnicas, pero la idea general de las estructuras y algoritmos utilizados se debe mantener.

En particular, se debe documentar si hay funciones que debieron ser agregadas a un módulo para facilitar el testeo (por ejemplo, una función para mostrar el contenido del módulo, o para serializarlo en una cadena, etc.). Dichas funciones extra, que gozan de ciertas libertades de visibilidad (public vs private) por razones técnicas, solo pueden usarse para testing.

Testing

Para poder hacer el testing, la cátedra facilitará un entorno que permite chequear el módulo principal. Este entorno consta de un programa que lee archivos de entrada y genera una salida única. La correctitud se constata comparando dicha salida con una salida correcta ya generada.

Para que este programa funcione, deben respetar las interfases provistas en los .h que se pueden bajar de la página. Una posibilidad es que renombren sus funciones para implementar directo sobre ese mismo módulo, y otra es que lo utilicen como un "wrapper", es decir, que implementen las funciones requeridas por el programa de testing como un simple llamado a una función de su propio módulo.

Luego de hecho esto, podrán linkearlo con el parser presentado, para que automáticamente puedan generar y probar lo realizado mediante casos de tests hechos como archivos de texto.

El formato de dichos archivos de texto es el siguiente: Los archivos tendrán nombres del tipo <nombre archivo>.in y el programa generará una salida en un archivo <nombre archivo>.out. Las instrucciones vienen dadas de la manera
<instrucción> <parámetro 1> <parámetro 2> ... <parámetro n >
Los parámetros siempre vienen separados por espacios, tanto de las instrucciones como entre ellos.

A continuación se muestra un archivo .in de ejemplo

```
nuevo 1
interfaces
agVersion 4
versiones
agEvento < CAIDA 12 0 >
eventos 0
tiempoCaida 0
agEvento < NOCAIDA 13 0 >
eventos 0
tiempoCaida 0
```

Para los parámetros, se cuenta con la siguiente sintaxis: Los conjuntos se delimitan con "{/}", las secuencias con "[/]", los arreglos con "(/)" y las tuplas con "</>".

Siempre comenzará el archivo con el comando "nuevo", que recibe un nat n y asume el conjunto de interfaces entre 0 y $n-1$.

A continuación mostramos algunas acciones y posibles salidas esperadas:

Generadores:

```
In: nuevo 4
Out: Nada

In: agVersion 6
Out: Nada

In: agRegla < ( 3 6 1 9 ) 1 5 >
Out: Nada

In: agEvento < CAIDA 1 4 >
Out: Nada
```

Observadores:

```
In: versiones
Out: { 3 4 6 }

In: interfaces
Out: 4

In: enrutar ( 3 2 1 4 )
Out: DireccionNoSoportada
Out: InterfazDeSalidaNoEncontrada
Out: InterfazDeSalidaCaida 4
Out: SalidaPorInterfaz 4

In: eventos 4
Out: [ < CAIDA 1 4 > < NOCAIDA 2 4 > ]
```

Nota: A hacer el operador << de los conjuntos, deben devolver en forma ordenada los elementos, para hacer única la salida.

Se recomienda además hacer tests para cada módulo individual. Estos no son soportados por el programa, pero pueden hacerlo con código de test, de manera similar a lo que se hizo en el TP Labo. Las funciones que implementan tests deben estar en archivos separados de las que implementan el módulo en sí, y deben estar diferenciados claramente los archivos de código o de encabezados correspondientes a la implementación de los correspondientes a tests.

Entrega

La entrega puede hacerse hasta 2 veces. Para entregar deben estar todos los integrantes del grupo presentes, tener el código implementado, el informe del TP3 y los informes corregidos de la o las entregas del TP2.

Se probará que su programa funcione correctamente utilizando el programa de testing provisto (una versión fresca, asi que asegúrense que funcione con la versión exacta que les proveemos y no la modifiquen) y se verificará que no pierda memoria. Si el programa funciona correctamente y no pierde memoria, se verificará que el código responda a lo diseñado en el TP2 (módulo lo que digan en el informe del TP3).

Si el programa presenta bugs (es decir, falla o pierde memoria en alguna cantidad no nula de casos) se les pedirá que los arreglen y se les volverá a corregir cuando lo hayan hecho. Dado esto, recomendamos asegurarse antes de entregar que su programa funciona y no pierde memoria, de manera de optimizar el tiempo.

Respecto de las fechas: Se **debe** entregar el TP por primera vez en al menos una de las 3 primeras fechas, completo y correcto respecto del TP2. La última fecha es simplemente el límite que tienen para terminar de corregir los errores o pérdidas de memoria, pero no se aceptará una entrega inicial ese dia. Si el TP es entregado sin alguna de las partes diseñadas, o profundamente incompleto (léase, no compila, no corre correctamente al menos en varios casos, etc) no se considerará una entrega completa y no se les acepatará como válida para permitirles entregar el miércoles 23.

Por otro lado, no todos los docentes estaremos presentes en todas las fechas, así que deben establecer con su corrector con cierta anticipación el día que vayan a presentar su TP. El corrector, salvo indicación contraria que intentaremos evitar, será el mismo que el del TP2.