ESCUELA DE CIENCIAS DE LA COMPUTACION E INFORMATICA
UNIVERSIDAD DE COSTA RICA

CI-1201 PROGRAMACION 2
I semestre del 2015
Prof. Alan Calderón Castro

Cuarto laboratorio guiado de programación

Objetivos

Al completar este laboratorio el estudiante debería ser capaz de:

1. Comprender y aplicar el esquema de trabajo “Controlador-Modelo”.
2. Comprender y usar plantillas de clase de la STL de C++.
3. Comprender e implementar un diseño de clases.
4. Elaborar plantillas de clase.
5. Realizar pruebas sobre plantillas de clase.

Descripción del problema

Se deberá construir un programa que realice una simulación del proceso de infección de computadoras como la mostrada en clase por el docente. La resolución del problema se deberá basar en el diseño provisto por el docente, lo que implica que los archivos de encabezados de las clases provistas por el docente NO PODRÁN SER MODIFICADOS A MENOS QUE ASÍ LO AUTORICE EXPLÍCITAMENTE.

De acuerdo con lo anterior usted deberá programar las siguientes clases:

Nombre de clase	¿qué representa?
Grafo	Representa la red de computadores sobre la cual se diseminará el virus. Se basará en componentes de la STL. LO NUEVO es que se deberá especializar la plantilla provista “GrafoGnr< T >” para generar esta clase.
Simulador	Representa el proceso de simulación de la infección en la red de computadoras.
Visualizador	Permite la visualización gráfica de una instancia de Grafo.

Además, se deberán programar:

1. Las pruebas de Grafo (programa aparte del principal).
3. El principal que será de consola y basado en “línea de comandos” y que responderá a las siguientes órdenes:

1. cargar “datosGrafo.txt” : que permitirá cargar los datos de un grafo desde un archivo de texto.
2. crear #vrt #prbAdy : que creará un grafo con cntVrt == #vrt en el que cada adyacencia (i,j) tenga la probabilidad de existir de prbAdy.
3. simular #cltr #ios #vsc #vcf #rc #grc : que simulará la infección (ver especificación) sin visualización de los estados intermedios, sólo la visualización de los estados inicial y final.
4. simular-visualizar #cltr #ios #vsc #vcf #rc #grc : que simulará la infección (ver especificación) visualizando todos los estados intermedios, el inicial y el final.

5. visualizar : que mostrará gráficamente el grafo previamente cargado o creado. Si no ha sido cargado ni creado el grafo, se mostrará la advertencia “DEBE CREAR O CARGAR GRAFO ANTES!!”
6. calcular-promedio-longitud-caminos-cortos : que calculará el promedio de las longitudes de los caminos más cortos del grafo previamente cargado o asociado. Si no ha sido cargado ni creado el grafo, se mostrará la advertencia “DEBE CREAR O CARGAR GRAFO ANTES!!”
7. calcular-centralidad-intermedial #vrt : que calculará la centralidad intermedial del vértice cuyo índice es #vrt (ver especificación) en el grafo asociado. Si no ha sido cargado ni creado el grafo, se mostrará la advertencia “DEBE CREAR O CARGAR GRAFO ANTES!!” NOTA: sigue pendiente para último laboratorio.
8. calcular-coeficiente-agrupamiento #vrt : que calculará el coeficiente local de agrupamiento para vértice cuyo índice es #vrt en el grafo asociado. Si no ha sido cargado ni creado el grafo, se mostrará la advertencia “DEBE CREAR O CARGAR GRAFO ANTES!!”

Es altamente recomendable que el presente laboratorio guiado sea desarrollado en parejas, no se podrán hacer grupos de más de dos personas.

Procedimiento

Este es un laboratorio guiado y por tanto la resolución del problema planteado deberá realizarse en el siguiente orden:

1 Elaborar un programa de pruebas para la plantilla de clase “GrafoGnr< T >”:
  1.1 Comprobar que el constructor **GrafoGnr< T >(int cntVrt, double prbAdy)**:
    1.1.1 Construye una red con la cantidad de vértices y el promedio de adyacencias por vértice correctos cuando cntVrt == 100 y prbAdy == 0,5.
    1.1.2 Construye una red con la cantidad de vértices y el promedio de adyacencias por vértice correctos cuando cntVrt == 1000 y prbAdy == 0,5.
  1.2 Comprobar que el constructor de copias **GrafoGnr< T >(const Grafo& orig)**:
    1.2.1 Construye una copia idéntica a Grafo(100,0.5).
    1.2.2 Construye una copia idéntica a Grafo(1000,0.5).
  1.3 Comprobar que el constructor **GrafoGnr< T >(string nArch):**
    1.3.1 Construye el grafo correcto con el “grafo_pequenyo.txt”.
    1.3.2 Construye el grafo correcto con el “grafo_grande.txt”.
  1.4
  1.5 Comprobar que “**double promLongCmnsCrts()**”:
    1.5.1 Genera el valor correcto con un grafo muy pequeño.
  1.6 Comprobar que **double coeficienteAgrupamiento(int vrt) const**.
    1.6.1 Genera el valor correcto para el vértice indicado de un grafo muy pequeño.
  1.7 Las pruebas anteriores deberán realizarse para un tipo básico como int (double o float) y para un tipo de vértice que incluya los atributos: tmpChqVrs, cntChqVrs y e (estado) que luego servirá para elaborar la simulación de dispersión de virus en una red de computadores.
  1.8 **PRIMER PUNTO DE CHEQUEO**.
2 Implementar la plantilla de clase “GrafoGnr< T >”:
  2.1 Implementar el constructor Grafo(int cntVrt, int prbAdy).
  2.2 Implementar el constructor de copias.
  2.3 Implementar el constructor Grafo(string nArch).

2.4 Implementar los métodos observadores básicos.
2.5 Implementar los métodos observadores especiales.
2.6 Implementar los métodos modificadores.
2.7 **SEGUNDO PUNTO DE CHEQUEO**.

3 Elaborar una clase concreta (no abstracta ni plantilla de clase) que incluya los atributos necesarios para la simulación de dispersión de virus. Esta clase además debe incluir la programación necesaria para calcular el nuevo estado de un vértice a partir de su estado actual y el de sus vecinos. **TERCER PUNTO DE CHEQUEO**.

4 Realizar los cambios necesarios sobre la clase Simulador:
4.1 Implementar el constructor Simulador(GrafoGnr<...>& g).
4.2 Implementar el método simular(…).
4.3 Se trata de una clase concreta, o sea: no es abstracta ni es plantilla de clase.
4.4 **CUARTO PUNTO DE CHEQUEO**.

5 Realizar los cambios necesarios sobre la clase Visualizador:
5.1 Implementar el constructor Visualizador(GrafGnr>...>o& g, Simulador& s).
5.2 Implementar el método visualizar().
5.3 Implementar el método visualizar(...).
5.4 Se trata de una clase concreta, o sea: no es abstracta ni es plantilla de clase.
5.5 **QUINTO PUNTO DE CHEQUEO**.

6 Realizar los cambios necesarios sobre el programa principal: proceder en el mismo orden que se indicaron las órdenes a las que deberá responder el programa principal. **SEXTO PUNTO DE CHEQUEO**.

7 **ENTREGA programada para el lunes 6 de julio a las 9:00am**.

8 **Para optar por el puntaje extra del quinto laboratorio**:
8.1 Elaborar una plantilla de clase abstracta “Simulador<...>” que facilite la elaboración de la clase Simulador de cualquier simulación específica.
8.1.1 Demostrar que funciona construyendo con esta plantilla la simulación de dispersión de virus.
8.1.2 Demostrar que funciona construyendo con esta plantilla la simulación de anidamiento sincronizado.
8.2 Elaborar una plantilla de clase abstracta “Visualizador<...>” que facilite la elaboración de la clase Visualizador de cualquier simulación específica.
8.2.1 Demostrar que funciona construyendo con esta plantilla la visualización de dispersión de virus.
8.2.2 Demostrar que funciona construyendo con esta plantilla la visualización de anidamiento sincronizado.
8.3 Elaborar una clase abstracta “VerticeGnr” que facilite la elaboración de la clase de vértice para cualquier simulación específica.
8.3.1 Demostrar que funciona heredando esta clase abstracta la clase de vértice necesaria para la simulación de dispersión de virus.
8.3.2 Demostrar que funciona heredando esta clase abstracta la clase de vértice necesaria para la simulación de anidamiento sincronizado.
8.4 Demostrar que los tres componentes anteriores funcionan correctamente integrándolos en las dos simulaciones: dispersión de virus y anidamiento sincronizado.

El seguimiento del laboratorio se hará en el aula de laboratorio directamente por el docente quien verificará la realización de cada uno de los pasos principales con cada pareja de trabajo conforme vayan avanzando.

Criterios de evaluación

La nota final de su trabajo dependerá de si en su programa se:

1. Respetan las reglas de estilo del código: márgenes, nombres de objetos empiezan en minúsculas, nombres de clases empiezan en mayúsculas, nombres de métodos también empiezan en minúscula pero se usan mayúsculas para concatenar palabras, comentarios para los atributos y variables de métodos.
2. Respeta la división de responsabilidades entre el programa controlador (que es el main()) y "Modelo", de manera que el "Controlador" se ocupa de la entrada de datos, todos los cálculos, el procesamiento de los datos y el despliegue de los resultados por la "Consola".
3. Ha simplificado el código lo más posible evitando el procesamiento duplicado de datos.
4. Se documentan mediante REQ, MOD y EFE los principales métodos (omita constructores, destructor, getters y setters).
5. Hace un uso óptimo de la memoria RAM en las estructura de datos siguiendo los lineamientos discutidos en clase para implementar las clases.

Productos

A través del sitio del curso, usted deberá entregar:

- Una carpeta comprimida, usando 7-zip, con los archivos de código fuente. El nombre del archivo comprimido y de la carpeta contenida se basará en el número de carnet de los participantes, por ejemplo "A12345_A67890_tp1.7z".
- En caso de que su programa no procese correctamente todas las operaciones descritas, haga un reporte de errores. Para cada error explique: 1) en qué consiste el error, 2) cuál cree usted que es la causa, 3) qué hizo para corregirlo, aunque no haya funcionado. En la medida en que este reporte muestre su dedicación al trabajo su nota final para esta tarea podría mejorar.

Fecha de entrega: **lunes 6 de julio a las 9:00 horas (4 semanas)**.

Evaluación:

1. Respeto de las reglas de estilo y simplificación del código:.....…….5%
2. División de responsabilidades entre "Controlador" y "Modelo":……..….5%
3. Efectividad y eficiencia de las estructura de datos y los algoritmos:.....…...90%
   - Pruebas de GrafoGnr< T>…..……...15%
   - Plantilla de clase GrafoGnr< T >……..25%
   - Clase Vértice……...20%
   - Clase Simulador……10%
   - Clase Visualizador……10%
   - Programa principal……10%
4. Hasta 10/100 puntos extra por un buen reporte de errores en caso de no lograrlo.

Notas:

1. Este proyecto deberá realizarse idealmente y a lo más en ***parejas***. **NO SE ACEPTARÁ NINGÚN TRABAJO ELABORADO POR MÁS DE DOS PERSONAS.**
2. **SI SU TRABAJO TIENE VIRUS, SERÁ PENALIZADO CON 20 puntos MENOS DE LA NOTA QUE SE LE ASIGNE.**
3. **A TODOS LOS ESTUDIANTES INVOLUCRADOS EN UN FRAUDE SE LES APLICARÁ EL ARTÍCULO #5 INCISO C DEL "Reglamento de Orden y Disciplina de los Estudiantes de la Universidad de Costa Rica".**