PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

LENGUAJES DE PROGRAMACIÓN 1

Examen 2
(Segundo Semestre 2017)

Indicaciones Generales:

- Duración: 3h.
- Se podrá usar como material de consulta solo sus apuntes de clase.
- No se pueden emplear variables globales, estructuras ni las funciones fopen, strtok, malloc, realloc, sscanf ni sprintf. La **clase string** solo se podrá emplear en donde expresamente se indique. Tampoco se podrá emplear la biblioteca estándar de plantillas (**STL**) salvo se indique lo contrario.
- La clausula **friend** solo se podrá emplear para sobrecargar los operadores de inserción y extracción de flujo (<< y >>), y en el caso de clases auto referenciadas para ligar el nodo con la estructura **inmediata** a ella.
- Deberá respetar estrictamente el encapsulamiento de datos en las clases a todo nivel.
- Los programas que presenten errores de sintaxis o de concepto se calificarán en base al 40% de puntaje de la pregunta. Los que no den resultados coherentes en base al 60%.
- Cada módulo no debe sobrepasar las 20 líneas aproximadamente.
- **La presentación, la ortografía y la gramática de los trabajos influirá en la calificación.**

Puntaje total: 20 puntos

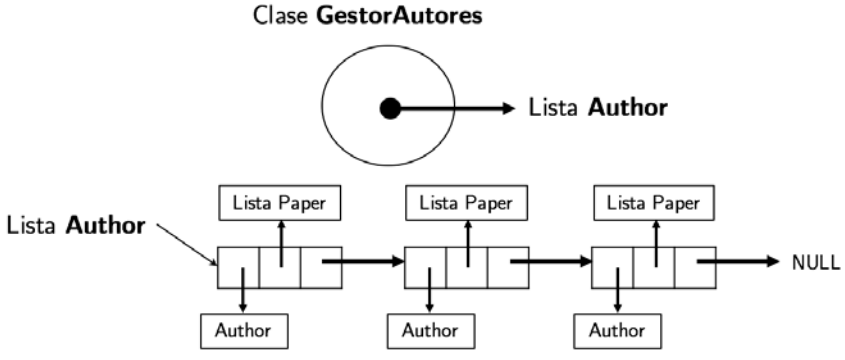
Cuestionario

Se desea elaborar una aplicación orientada a objetos que permita controlar y adminsitrar la información de publicaciones académicas. Para ello, se trabajarán con diferentes estructuras para probar distintos enfoques de manipulación y encapsulamiento en contenedores secuenciales usando clases y objetos.

PREGUNTA 1 (2 puntos)
Implemente las clases **Paper**, **ConferencePaper**, **JournalPaper** y **Author**. Estas clases deben poseer los atributos y métodos ya conocidos previamente.

OJO: Las preguntas 2, 3 y 4 son independientes entre sí, no hay un orden explícito para resolverlas, y todas pueden empezar con las clases sin modificación de la Pregunta 1.

PREGUNTA 2 (4.5 puntos)
Elabore un proyecto denominado "**Pregunta02**", copie en él las clases desarrolladas en la Pregunta 1 e incorpórelas al proyecto.
Gestor de Autores: implemente una clase gestora que posea una lista de Autores que adjunte la lista de autores de la siguiente forma:



Tome en cuenta que cada nodo de la lista tiene 3 atributos: un objeto Author, una lista de Papers y el puntero al siguiente nodo.
Para ambas listas (Author y Paper) debe implementar clases propias (incluyendo nodos). No es obligatorio que use clases plantillas, pero debe considerar los siguientes métodos:

- Un método de inserción ordenada según el operador "<" del objeto (Author o Paper).
- Un método de impresión de toda la lista.
- Un método para eliminar toda la lista.

Por otro lado, la clase de gestión de autores debe contar con los siguientes métodos:

- Método Constructor por defecto y Destructor que considere la creación y eliminación de la lista.
- Un método que lea datos de diversas publicaciones desde un archivo (considerar el archivo de prueba "Authors.csv") y populе la lista de **Author**.
- Un método que lea datos de diversas publicaciones desde un archivo (considerar el archivo de prueba "Papers.csv" con el caracter inicial por línea: "C" o "J") y populе la lista de **Paper** en el nodo del **Author** correspondiente. Implemente los métodos auxiliares que considere necesarios en el gestor o en la lista. Asuma que este método se invocará cuando el archivo de autores ya haya sido procesado.

En el archivo principal ("main.cpp") debe realizar una prueba instanciando un objeto de la clase **GestorAutores**, el cual debe leer los archivos de prueba, imprimir la lista completa (**CITANDO** los objetos Paper) y destruirse al final del programa. No debe instanciarse ninguna clase del tipo lista en este ámbito.

PREGUNTA 3 (4.5 puntos)
Elabore un proyecto denominado "**Pregunta03**", copie en él las clases desarrolladas en la Pregunta 1 e incorpórelas al proyecto.
Gestor de Publicaciones: implemente una clase gestora que posea una lista de publicaciones (**Paper**) que adjunte la lista de publicaciones de la siguiente forma:

En este caso, se debe hacer una **modificación** sobre Paper, ya que en realidad estos son desarrollados **por más de un autor**. Por ello, debe **reemplazar** el atributo entero de "id_author" por una **lista STL** de objetos tipo **Author**.
Debe actualizar los métodos necesarios en la clase Paper y derivados, para que pueda trabajar con la información de varios autores. Vea como ejemplo el archivo "**Papers02.csv**", donde la información de los múltiples autores se encuentra en el mismo archivo. Después del tipo ("C" o "J") y el "id_paper", se recibe el **número de autores** (número entero), y luego se recibe el "id_author", "first_name", y "last_name" según la cantidad indicada. Posteriormente, se procesa la información de la publicación como ya se conoce.

Para la lista Paper, debe implementar clases propias (incluyendo nodos). No es obligatorio que use clases plantillas, pero debe considerar los siguientes métodos:

- Un método de inserción ordenada según el operador "<" del objeto (Paper).
- Un método de impresión de toda la lista.

- Un método para eliminar toda la lista.

Por otro lado, la clase de gestión de autores debe contar con los siguientes métodos:

- Método Constructor por defecto y Destructor que considere la creación y eliminación de la lista.
- Un método que lea datos de diversas publicaciones desde un archivo (considerar el archivo de prueba "Papers02.csv" con el caracter inicial por línea: "C" o "J") y populе la lista de **Paper** con los múltiples objetos **Author** como su atributo (lista STL).

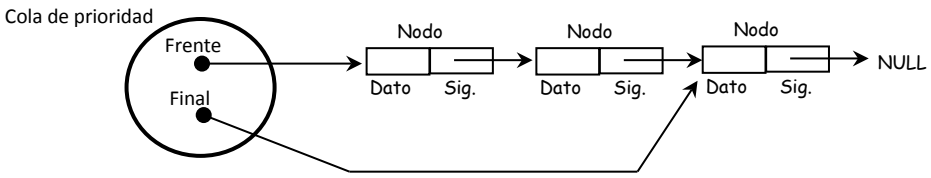
En el archivo principal ("main.cpp") debe realizar una prueba instanciando un objeto de la clase **GestorPublicaciones**, el cual debe leer el archivo de prueba, imprimir la lista completa (***CITANDO*** las publicaciones con múltiples autores) y destruirse al final del programa. No debe instanciarse ninguna clase del tipo lista en este ámbito.

PREGUNTA 4 (9 puntos)

Una cola de prioridad (priority queue) es una estructura de datos en la que los elementos se atienden en el orden indicado por una prioridad asociada a cada elemento. Si varios elementos tienen la misma prioridad, se atenderán de modo convencional dependiendo del momento de llegada a la cola. Por ejemplo si llegaran los siguientes pares de números: (2,3), (4,1), (2,1), (3,3), (1,1), (3,1), (2,4), (1,8) y (3,5), y la prioridad estuviera dada por el primer elemento del par, los pares serían atendidos en el siguiente orden: (4,1), (3,3), (3,1), (3,5), (2,3), (2,1), (2,4), (1,1), (1,8).

Se desea elaborar **plantilla de clases** que permita crear y manejar una cola de prioridad (PQ). No podrá emplear en esta pregunta contenedores STL.

La plantilla que maneje esta PQ se basará en el siguiente esquema:



La plantilla que maneje la PQ deberá manejarse a través de dos atributos (punteros): "**frente**" que apuntará al elemento de la cola que será atendido y "**final**" que apuntará al último elemento de la cola.

La plantilla que maneje el nodo de la cola deberá manejarse a través de dos atributos: "**dato**" que representa la clase que definirá el objeto que se formará en la PQ y "**siguiente**" que apuntará al siguiente nodo de la PQ.

La PQ deberá implementar, además de los constructores y destructores, los siguientes métodos:

Arribo: llega un elemento a la PQ y es ubicado en la cola según su prioridad.

Atención: el nodo del frente de la PQ es extraído, eliminándolo de la cola y entregando una copia del dato.

Listado: muestra los datos contenidos en la PQ, los datos no son extraídos de la cola. El listado debe aparecer en un archivo de textos (no re direccionando la salida estándar), con títulos adecuados y con los datos muy bien tabulados (sin usar el carácter '/t') de modo que se entienda fácilmente la información contenida en él.

Finalmente deberá desarrollar dos proyectos (**PruebaPQ1** y **PruebaPQ2**), en el primero deberá probar la plantilla empleando una clase "**ParEnteros**" que contenga como atributos dos valores enteros y que la prioridad la tenga el primer entero. Esta clase deberá definir constructores, destructores, métodos selectores, y todos los métodos que se requieran para poden ser manipulados por la PQ. En el segundo proyecto se probará la PQ con la clase **Author** que tenga como prioridad número de papers (int).

CONSIDERACIONES FINALES:

- Cree en el computador una carpeta de trabajo con la siguiente ruta: c:\temp\Examen2. En ella colocará los proyectos que den solución al problema planteado.
- **En cada archivo que implemente en los proyectos (.h y .cpp) deberá colocar un comentario en el que coloque claramente su nombre y código. De no hacerlo se le descontará 0.5 puntos por archivo.**
- De no respetarse el nombre de los proyectos se descontará 1 punto por cada trasgresión.
- La calificación se otorgará por proyecto desarrollado. Por ninguna razón se asignará puntaje a dos o más preguntas por el mismo proyecto.

NO SE HARÁN EXCEPCIONES

Al finalizar el examen, comprima[1] la carpeta ***Examen2*** en un archivo con nombre <código del alumno con 8 dígitos>.zip y súbalo a la intranet del curso, en el enlace Documentos, en la carpeta \Examen2\<código del horario>\<aula>. El acceso a la Intranet quedará cerrado automáticamente a las 11:05 am. por lo que el alumno que no suba alguno de los proyectos a la Intranet recibirá como nota CERO en esa pregunta. **NO SE HARÁN EXCEPCIONES**.

Profesores del curso: Arturo Oncevay
Miguel Guanira

San Miguel, 5 de diciembre del 2017.

[1] Para evitar problemas en la corrección de la prueba, utilice el programa de compresión que viene por defecto en el Windows (Zip) **no use 7z**.