Se evaluará la eficiencia del código y las buenas prácticas de programación como: el uso de identificadores significativos, la indentación, escritura correcta de llaves {} y el uso adecuado de la palabra reservada const. Se dispone de tres horas para entregar la prueba y debe realizarse en forma estrictamente individual.

Una compañía debe trabajar con miles de archivos XML. Muchos de ellos fueron creados sin cuidado y son causa de frustración cuando un empleado debe analizarlos. El dueño le ha contratado para implementar un conjunto de rutinas eficientes para procesamiento de archivos XML. La primera que le han solicitado es un comando de Unix para indentar correctamente archivos XML. El programa debe recibir una lista de nombres de archivo por parámetro, abrilos uno a uno, indentarlos, y mostrar el resultado indentado en la salida estándar. Por ejemplo, si su programa se invoca con el siguiente inventario de la ECCI:

```
<inventario nombre="ecci"><activo tipo="edificio"><activo
tipo="aula" nombre="IF103"><activo tipo="escritorio"
placa="10929"/><activo tipo="video-beam" placa="2189019">
Mal ubicado. Acercarlo a la pizarra</activo><activo
tipo="computadora" nombre="HP119" placa="2190132"
estado="dañado"><activo tipo="disco-duro" nombre="WD899210"
estado="dañado">Tiene sectores dañados. Reemplazar.</activo>
</activo></activo>
<!-- Falta el resto de aulas, oficinas y laboratorios -->
</activo></inventario>
```

debe producir un resultado similar al siguiente:

```
<inventario nombre="ecci">
   <activo tipo="edificio">
      <activo tipo="aula" nombre="IF103">
         <activo tipo="escritorio" placa="10929"/>
         <activo tipo="video-beam" placa="2189019">
            Mal ubicado. Acercarlo a la pizarra
         </activo>
         <activo tipo="computadora" nombre="HP119" placa="2190132" estado="dañado">
            <activo tipo="disco-duro" nombre="WD899210" estado="dañado">
               Tiene sectores dañados. Reemplazar.
            </activo>
         </activo>
      </activo>
      <!-- Falta el resto de aulas, oficinas y laboratorios -->
   </activo>
</inventario>
```

Para poder indentar un archivo XML, el programa debe reconocer su estructura. Un documento XML se construye con 12 tipos de estructuras llamadas en forma genérica **nodos**. Para esta evaluación, su programa sólo debe trabajar con tres de estos 12 tipos de nodos:

1. **Elemento**. Conforman las partes lógicas del documento, por ejemplo, un libro se compone de capítulos, secciones, párrafos, ilustraciones, etc. Cada una de estas partes es un elemento. En XML tienen la forma `<elemento atributo="valor">contenido</elemento>`. El contenido es opcional, y en tal caso, el elemento se puede escribir `<elemento atributo="valor"/>`. Un elemento puede tener cero o más atributos de la forma `atribut="valor"`. El valor siempre debe estar dentro de comillas (puede asumir que siempre son comillas dobles).
2. **Comentario**. Tienen la forma `<!-- texto del comentario -->`. El texto puede ser de cualquier longitud hasta encontrar los caracteres `-->`.
3. **Texto**. Es texto puro que se encuentra como parte del contenido de los elementos. Por ejemplo `"Tiene sectores dañados. Reemplazar."`.

En un documento XML los nodos se organizan jerárquicamente, siendo la raíz un elemento. Cada nodo tiene una cantidad arbitraria de nodos hijos, los cuales pueden ser de naturaleza muy heterogénea (sugerencia: utilice un arreglo para referirlos). Un nodo además tiene un entero que indica el tipo de nodo que es: elemento, comentario o texto (sugerencia: utilice una enumeración). Finalmente, todo nodo tiene un nombre (por convención llamado `nodeName`), cuyo valor depende del tipo de nodo.

Los nodos **elemento** utilizan como nombre (`nodeName`) el mismo que se cargó del archivo, por ejemplo, "`inventario`" y "`activo`". A diferencia de los demás nodos, los elementos tienen una lista de atributos (sugerencia: puede utilizar un `std::map` para representarlos), los cuales son parejas de dos cadenas de caracteres. El orden de los atributos no importa en el documento, pero es prohibido tener dos o más atributos con el mismo nombre.

Los nodos **comentario** no tienen nombre (`nodeName`) en el documento. Por convención se utiliza "`#comment`". No tienen atributos pero sí una cadena de texto que representa su valor. Lo mismo ocurre con los nodos de **texto**, y su nombre por convención es "`#text`".

Implemente una clase `Documento`, la cual tiene un método que recibe un nombre de archivo y carga de él una jerarquía de nodos, y otro método para imprimir el documento en un archivo o la salida estándar. La impresión debe ser perfectamente indentada. Usted debe sobrescribir los operadores de flujo >> y << para hacer la carga e impresión del documento. Puede asumir que los documentos no tienen errores sintácticos, lo que en nomenclatura XML se conoce como "bien formados".

Evaluación

1. [20%] Representa correctamente la jerarquía de nodos. Los nodos conocen su tipo, nombre e hijos. La jerarquía evita fugas a toda costa. La clase Nodo es abstracta (un nodo "puro" no sabe cómo cargarse o imprimirse).
2. [15%] Representa correctamente los nodos elemento. Los elementos tienen atributos. Los elementos se cargan e imprimen correctamente en archivos.
3. [20%] Representa correctamente los nodos de texto y comentarios. Se cargan e imprimen adecuadamente de los archivos.
4. [15%] Representa correctamente el documento con una clase `Documento`. El documento conoce su jerarquía de nodos. Evita fugas de memoria a toda costa. Sobrecarga y utiliza los operadores de flujo (>> y <<). Utiliza esta clase adecuadamente desde el `main()`.
5. [15%] Utiliza la jerarquía de nodos adecuadamente para cargar el documento desde el archivo.
6. [15%] Utiliza la jerarquía de nodos adecuadamente para imprimir los nodos con indentación perfecta en la salida estándar.