Programación 4 - 2025
Laboratorio 0 :: Conceptos Básicos en C++

Consideraciones generales:

- La entrega podrá realizarse hasta el **30 de marzo de 2025 a las 23:59hrs.**
- El código fuente y el archivo Makefile [1] deberán ser entregados mediante el EVA del curso [2] dentro de un archivo con nombre `<número de grupo>_lab0.zip (o tar.gz)`.
- Más allá de que se sugiere el uso de un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para contar con un ambiente de desarrollo adecuado, el archivo Makefile entregado debe ser independiente de cualquier IDE, permitiendo la compilación aislada de la solución.
- El código deberá poder compilarse y ejecutarse sin errores en las máquinas Linux de la Facultad de Ingeniería **con la flag del compilador -std=c++98[1] [3] (no estando permitido el uso de auto, range-based loops, etc. en los archivos definidos)**.
- Las entregas que no cumplan estos requerimientos no serán consideradas. El hecho de no realizar una entrega implica la insuficiencia del laboratorio completo.

Objetivo

El objetivo del Laboratorio 0 es reforzar conceptos básicos de orientación a objetos a través de su implementación en el lenguaje C++ [4], así como practicar diversas construcciones del lenguaje y del entorno de programación en Linux [5] que serán de utilidad en etapas posteriores del laboratorio. Se espera que se consulte el material disponible en el EVA del curso (ver referencias al final de este documento), complementado con consultas a otros medios (ej.: Internet) con espíritu crítico y corroborando que las fuentes consultadas sean confiables.

Problema

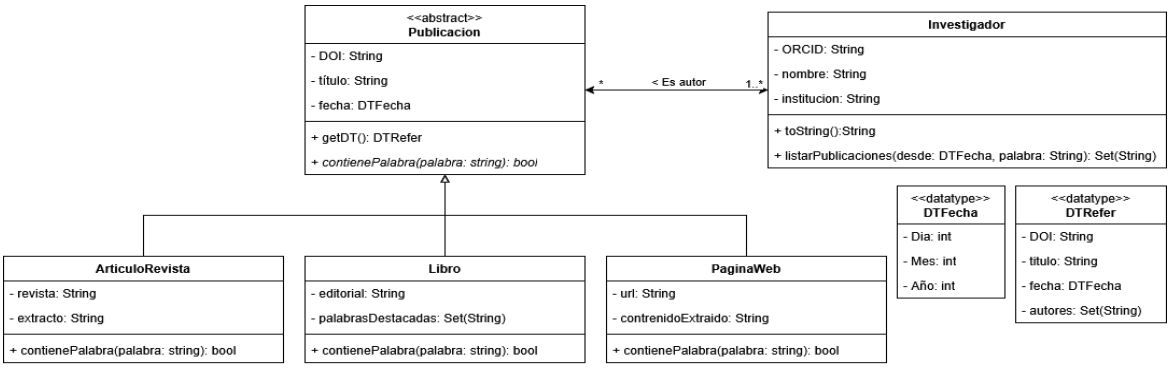
El diagrama de clases UML de la siguiente página representa el diseño de un sistema para la gestión de publicaciones académicas y sus autores. En este sistema, se modelan distintos tipos de publicaciones científicas y técnicas, así como a los investigadores que las crean, con el propósito de organizar y mantener un registro estructurado y centralizado de publicaciones y sus autores.

Para representar las publicaciones, el sistema define una jerarquía de clases en la que la clase abstracta `Publicacion` funciona como la superclase de tres tipos específicos: `ArticuloRevista`, `Libro` y `PaginaWeb`. Cada publicación cuenta con un DOI (Identificador de Objeto Digital), un string único que garantiza un enlace permanente al contenido digital, incluso si su ubicación cambia. Además, se almacena el título y la fecha en la que se registró la publicación. Dependiendo del tipo de publicación, se incluyen atributos adicionales. En un artículo de revista, se registra el nombre de la revista donde fue publicado y un extracto del artículo. En el caso de un libro, se almacena el nombre de la editorial y una lista de palabras destacadas asociadas a su contenido. Para una página web, se registra la URL donde

[1] Un ejemplo de llamado al compilador podría ser g++ -std=c++98 -o mi_programa main.cpp

está publicada y el contenido extraído de la misma. Todas estas propiedades se representan como atributos privados dentro de sus respectivas clases.

Los investigadores que elaboran estas publicaciones están representados por la clase `Investigador`, que incluye un identificador único ORCID (Open Researcher and Contributor ID) que consiste en un string de 16 dígitos (ejemplo, *0000-0001-2345-6789*) que permite asociar sus contribuciones científicas a su perfil personal. También se almacena su nombre e institución.



El sistema modela la relación entre `Investigador` y `Publicacion` mediante una asociación denominada *"es autor"*, que permite representar qué publicaciones han sido escritas por cada investigador. Asimismo, en la clase `Publicacion` se registran sus autores. En ambos casos se utiliza un pseudo-atributo para almacenar la colección de objetos asociados.

A nivel de comportamiento, se definen las siguientes operaciones:

- La operación `getDT()` en la clase `Publicacion` devuelve un datavalue de tipo `DTRefer`, que incluye el DOI, el título y la fecha del objeto, así como un listado (autores) de los nombres de los investigadores relacionados.

- La operación `contienePalabra(palabra: String)` se encuentra en las clases `Publicacion`, `ArticuloRevista`, `Libro` y `PaginaWeb`. Se trata de una operación polimórfica que devuelve un valor booleano, indicando si el objeto con el cual se invoca contiene o no la palabra especificada. En el caso de `ArticuloRevista` y `PaginaWeb`, la operación busca si el extracto o el contenido extraído contienen la palabra indicada. En el caso de `Libro`, se valida si la palabra es una de las palabras destacadas asociadas. Esta operación en abstracta en `Publicacion` y sus métodos están definidos en las clases `ArticuloRevista, Libro` y `PáginaWeb`.

- La operación `listarPublicaciones(desde: DTFecha, palabra: String)` de la clase `Investigador` obtiene el identificador de todas sus publicaciones que contengan la palabra indicada y cuya fecha sea posterior al parámetro `desde`. Para ello, se itera sobre el conjunto de publicaciones asociadas al investigador. Para cada publicación cuya fecha sea posterior a la indicada en el parámetro `desde`, se llama a la operación `contienePalabra`. Si esta devuelve verdadero, se invoca el getter del atributo DOI y se acumulan estos valores en un conjunto, que se retorna al finalizar la iteración.

- La operación `toString()` en la clase `Investigador` devuelve un string de la forma: `ORCID->nombre/institucion`.

Se pide:

1. Implementar todas las clases (incluyendo sus atributos, pseudo-atributos, constructores, destructores, getters y setters que sean necesarios) y datatypes que aparecen en el diagrama.

2. Sobrecargar el operador de inserción de flujo (<<) en un objeto `std::ostream`. Esta sobrecarga debe permitir imprimir todos los datos del datatype `DTRefer` siguiendo el siguiente formato: `DOI->titulo(fecha)/autor1,autor2...,autorN`

3. Implementar la operación `contienePalabra` en las clases `Publicacion`, `ArticuloRevista`, `Libro` y `PaginaWeb`.

4. Implementar la operación `toString` en la clase `Investigador`.

5. Implementar la operación `listarPublicaciones` en la clase `Investigador`.

6. Implementar un método `main` que defina un conjunto de `Publicacion` y un conjunto de `Investigador` y que ejecute la siguiente secuencia de pasos, donde se especifica cuáles deben imprimirse en la consola. Para evitar problemas en la salida en consola, se eliminaron los tildes y la ñ en los datos.

    a. Crear los siguientes objetos de la clase `ArticuloRevista` (con el constructor por parámetros):

DOI	10.1234/abc123
titulo	Fundamentos de POO
fecha	15/5/2023
revista	Programación Avanzada
extracto	Introduccion a los principios fundamentales de la programacion orientada a objetos, explicando sus conceptos clave como clases, objetos, herencia y polimorfismo.

DOI	10.4567/jkl012
titulo	Utilidad de diagramas UML
fecha	10/2/2024
revista	Modelado de Software
extracto	Ejercicio empirico de como los diagramas UML pueden ayudar en el proceso y documentacion de software, cubriendo los tipos mas importantes utilizados, como clases.

b. Crear los siguientes objetos de la clase `Libro` (con el constructor por parámetros):

DOI	10.2345/def456
titulo	Patrones de Diseno en c++
fecha	20/8/2022
editorial	Software Design
keyWords	Diseno OOP Class

DOI	10.5678/mno345
titulo	Guia de UML
fecha	20/8/2022
editorial	IEEE
keyWords	Diagramas UML Software Modelado

c. Crear los siguientes objetos de la clase `PaginaWeb` (con el constructor por parámetros):

DOI	10.3456/ghi789
titulo	Diagramas para Principiantes
fecha	20/10/2024
url	www.umlparaprincipiantes.com
contenidoExtraido	En esta pagina web se presenta una gui completa sobre los diagramas UML, abordando los diagramas de casos de uso, de clases, de secuencia y de actividades.

d. Imprimir en consola utilizando la inserción de flujo el resultado de ejecutar la operación `getDT()` para cada uno de los objetos `Publicacion` creados.

e. Crear los siguientes objetos de la clase `Investigador` (con el constructor por parámetros):

ORCID	0000-0003-1234-5678
titulo	Carla Oliveri
institucion	Universidad de la Republica

ORCID	0000-0001-8765-4321
titulo	Alberto Santos
institucion	Instituto Tecnico

f. Imprimir en consola el resultado de ejecutar la operación `toString` para cada uno de los objetos `Investigador` creados.

g. Registrar las siguientes relaciones entre investigadores y publicaciones (creando links de la relación en ambas direcciones).

Investigador	Publicacion
0000-0003-1234-5678(Carla Oliveri)	10.1234/abc123 (Fundamentos de POO)
0000-0003-1234-5678 (Carla Oliveri)	10.4567/jkl012 (Utilidad de diagramas UML)
0000-0003-1234-5678 (Carla Oliveri)	10.5678/mno345 (Guía de UML)
0000-0003-1234-5678 (Carla Oliveri)	10.3456/ghi789 (Diagramas para Principiantes)
0000-0001-8765-4321 (Alberto Santos)	10.1234/abc123 (Fundamentos de POO)
0000-0001-8765-4321 (Alberto Santos)	10.2345/def456 (Patrones de Diseno en c++)
0000-0001-8765-4321 (Alberto Santos)	10.4567/jkl012 (Utilidad de diagramas UML)

h. Invocar la operación `listarPublicaciones(10/12/2023,"UML")` para la investigadora 0000-0003-1234-5678 (Carla Oliveri) e imprimir el resultado en consola (un string por línea).

i. Ejecutar la eliminación del objeto 10.4567/jkl012 (Utilidad de diagramas UML) de la clase `Publicacion`.

j. Invocar la operación `listarPublicaciones(1/1/2020,"UML")` para la investigadora 0000-0003-1234-5678 (Carla Oliveri) e imprimir el resultado en consola (un string por línea).

k. Imprimir en consola utilizando la inserción de flujo el resultado de ejecutar la operación `getDT()` para cada uno de los objetos `Publicacion` creados (mismo código que la parte d).

Notas:

- Puede implementar operaciones auxiliares en las clases dadas en el diagrama si considera que facilitan la resolución de las operaciones pedidas.
- Se puede utilizar el tipo `std::string` [6] para implementar los atributos de tipo string, así como estructuras de datos de la biblioteca STL [7], tales como vector, set, map, etc.
- Se deben solucionar los problemas de dependencias circulares entre las clases, por ejemplo con relaciones bidireccionales. Para ello es necesario utilizar declaraciones en avanzada (forward declarations) [8].

Referencias

[1] Programación 4. Instructivo de Compilación
URL: https://eva.fing.edu.uy/course/view.php?id=413&section=3

[2] EVA Programación 4.
URL: https://eva.fing.edu.uy/course/view.php?id=413

[3] C++98 Standard en GCC
URL:https://gcc.gnu.org/projects/cxx-status.html#cxx98

[4] C++.
URL: https://www.cplusplus.com/

[5] Unidad de Recursos Informáticos, Salas Linux.
URL: https://www.fing.edu.uy/es/sysadmin/ensenanza/salas-linux

[6] Tipo `std::string`
URL: https://en.cppreference.com/w/cpp/string/basic_string

[7] C++ Standard Template Library (STL)
URL: https://cplusplus.com/reference/stl/

[8] Programación 4. Referencias Circulares y Namespaces
URL: https://eva.fing.edu.uy/course/view.php?id=413&section=3