Ingeniería Civil en Computación e Informática
Ingeniería en Tecnologías de Información

Taller N.º 1

Estructuras de datos – 2° Semestre 2025 – José Luis Veas y Bastián Ruiz
Fecha del enunciado: 29/08/2025 Fecha entrega taller:12/09/2025

El objetivo de este taller es diseñar e implementar un sistema de gestión académica, utilizando listas enlazadas como estructura de datos principal. Con este proyecto se evalúa la comprensión y aplicación de conceptos de programación en C++ y el manejo de estructuras de datos dinámicas. El código del proyecto deberá ser gestionado a través de un repositorio en GitHub para asegurar un control de versiones adecuado.

Descripción del Proyecto

Se le ha solicitado a usted la construcción de un sistema de gestión de información académica para la universidad, donde se trata la información de los alumnos, cursos y notas. Este sistema debe implementar las siguientes funcionalidades:

- **Gestión de alumnos**: Registro, búsqueda y eliminación de alumnos.
- **Gestión de cursos**: Creación, búsqueda y eliminación de cursos.
- **Inscripción de cursos**: Inscribir a alumnos en cursos. También se debe poder suprimir la inscripción a estos.
- **Gestión de notas**: Asignación de notas a alumnos en cursos específicos.
- **Reportes**: Consultar la información de un alumno (cursos inscritos y notas) y calcular promedios de notas por curso y el promedio general del alumno.

Los alumnos del sistema contienen un identificador único, nombre, apellido, carrera y fecha de ingreso a la universidad. Los cursos tienen un código único, nombre, cantidad máxima de estudiantes, carrera y nombre del profesor que dicta el curso. Las notas que se registran se basan en el sistema chileno (de 1,0 a 7,0), y los alumnos pueden tener múltiples notas en un curso. Cabe destacar que un curso puede ser inscrito solo por los estudiantes de la misma carrera. Además, un alumno puede estar inscrito en más de un curso, como un curso puede tener muchos alumnos.

Requisitos

Requisitos Funcionales

En base a lo descrito anteriormente, el sistema debe cumplir con los siguientes requerimientos:

- Menú principal:
  - El sistema debe presentar un menú de opciones claro para la interacción del usuario.
- Manejo de Alumnos:
  - Crear un alumno con todos sus datos necesarios.
  - Buscar un alumno por ID o nombre y listar su información.
    - Si el nombre se repite, se debe listar la información de cada alumno cuyo nombre coincida con la búsqueda.
  - Eliminar un alumno del sistema utilizando su ID.
    - Al eliminar un estudiante, se debe eliminar su registro de notas e historial de cursos inscritos.
- Manejo de Cursos:
  - Crear un curso con todos sus datos necesarios.
  - Buscar curso por código o nombre.
    - Si el nombre se repite, se debe listar la información de cada curso cuyo nombre coincida con su búsqueda.
  - Eliminar un curso del sistema utilizando su ID.
    - Al eliminar un curso, se debe eliminar la inscripción de los estudiantes a dicho curso.
- Manejo de Inscripciones:
  - Inscribir un alumno en un curso.
  - Eliminar un alumno de un curso.
- Manejo de Notas:
  - Registrar una o más notas para un alumno presente en un curso.
    - Las notas deben ser valores numéricos según el sistema chileno (1,0 a 7,0).
- Consultas y Reportes:
  - Obtener todos los alumnos de una carrera.
  - Obtener todos los cursos en los que un alumno está inscrito.
  - Calcular el promedio de notas de un alumno en un curso.
  - Calcular el promedio general de un alumno (según sus promedios finales).

Requisitos Técnicos

- El sistema debe desarrollarse en el **lenguaje de programación C++**.
- El sistema debe funcionar mediante **consola de comandos**.
- El sistema debe incluir el uso de **listas enlazadas** (listas con nexo) de forma obligatoria para la organización de los datos. La implementación de estas debe realizarse de forma manual, sin utilizar las librerías nativas de contenedores incluidas en el compilador de C y C++.
- El proyecto debe gestionarse utilizando el sistema de control de versiones de **GitHub**. Debe crear un **repositorio público** y subir los cambios en la rama principal del proyecto. Se espera un **historial de commits coherente** con el desarrollo del proyecto.
- El código del proyecto debe seguir los **lineamientos de codificación en C++**. Este debe estar organizado de forma modular y aplicar la separación del código en archivos de encabezado (.h) y de implementación (.cpp).
- El proyecto debe **compilarse sin errores** con un compilador estándar (C++11 en adelante).
- La aplicación debe **validar las acciones del usuario**, manejando errores de entrada y condiciones de borde.

Pauta de Evaluación

El taller será evaluado en base a la siguiente pauta de cotejo.

Criterio	Descripción	Ponderación
Cumplimiento de Funcionalidades	El sistema debe implementar todas las funcionalidades listadas en los requisitos.	**30%**
Uso de Estructuras de Datos	Se evaluará la correcta implementación y manipulación de las listas enlazadas para la gestión de los datos.	**40%**
Calidad y Estilo de Código	Se considerará la legibilidad del código, la modularidad y el uso de buenas prácticas de programación.	**15%**
Gestión de Repositorio	Se evaluará el uso de Git y GitHub, incluyendo el historial de commits y la organización del repositorio.	**15%**

Reglas del Taller

- **Fecha de Entrega**: El proyecto debe ser entregado a más tardar el 12/09/25. Se debe entregar en Campus Virtual el enlace al repositorio público de GitHub. Se tomará en cuenta la fecha del último commit en la rama principal. **No se aceptarán entregas tardías por correo electrónico**.
  - El proyecto solo será revisado si se encuentra en un repositorio. Si este no es accesible (si se encuentra privado) o se entrega en otro formato (como un archivo comprimido), el taller **no será revisado**, lo que implicará su calificación con nota mínima (1,0).
- **Involucrados:** El taller puede ser resuelto de forma individual o en parejas. En el caso que se desarrolle de a 2 integrantes, se considerará en la evaluación del repositorio una cantidad equitativa de commits entre ambos alumnos. Se recomienda trabajar con ramas para evitar conflictos de integración, recordando que solo se evaluará lo que se encuentre en la rama principal del repositorio.
- **Originalidad**: El trabajo debe ser original y desarrollado por el estudiante. La detección de plagio o copia de código (ya sea de internet o de otro compañero) resultará en una calificación de nota mínima (1,0) para todos los involucrados. Se utilizarán herramientas de detección de plagio de código.
- **Comunicación**: Cualquier duda o problema técnico debe ser consultado en clase o a través de los canales establecidos.