



## Estructuras de Datos (2022-1)

### Laboratorio 4

*Profesor: Alexander Irribarra*

*Ayudantes: Leonardo Aravena, Diego Gatica, Vicente Lermenda*

## Objetivos

Los objetivos del laboratorio son:

- Mejorar la programación, compilación y ejecución de programas escritos en lenguaje *C++* u otros.
- Implementar distintas estructuras de datos.
- Analizar y comparar diferentes implementaciones de un mismo tipo de dato abstracto.

## Ejercicios

1. Crear el **ADT** (tipo de dato abstracto) **List** en una clase que tenga el mismo nombre (Debe estar contenido en el fichero *ListADT.h*). Esta interfaz debe contener los siguientes métodos como mínimo:
  - **Insertar elemento al principio:** `virtual void insert(int)=0;`
  - **Eliminar al final:** `virtual void pop()=0;`
  - **Acceder al *i*-ésimo elemento:** `virtual INT at(int)=0;`
  - **Obtener la cantidad de elementos almacenados:** `virtual int size()=0;`
2. Implementar la estructura de datos **ArrayList**, que debe heredar de la clase **List** y contener sus métodos implementados (Debe estar definido en los ficheros *ArrayList.h* y *ArrayList.cpp*). **Cuando un ArrayList supera la cantidad de elementos que puede almacenar, este debe duplicar la memoria que tiene asignada usando *malloc* o *new* y copiar los elementos previamente insertados en el nuevo espacio. No olvidar liberar la memoria previamente asignada con *free***

o *new*, dependiendo de cual función se utilizó para asignar la memoria.<sup>12</sup>

3. Implementar la estructura de datos **LinkedList**, que debe heredar de la clase **List** y contener sus métodos implementados (Debe estar definido en los ficheros *LinkedList.h* y *LinkedList.cpp*).
4. Se debe realizar un análisis experimental de las estructuras de datos implementadas, midiendo el tiempo promedio de cada uno de los métodos. Para probar los métodos *insert* y *pop* se debe tomar el tiempo promedio de **insertar/remover un elemento**, ejecutando *n* veces el métodos. Por otro lado, para el método *at*, se deben **buscar** elementos cuando ya hay *n* elementos insertados en la lista. Escribir los resultados experimentales en una tabla y realizar gráficos comparativos por cada método. Recuerda tomar valores de entrada equidistantes entre sí para una mejor apreciación de la complejidad.
5. ¿Cuál cree que es la mejor implementación para **ADT List**? ¿Por qué?

## Observación

Los estudiantes pertenecientes al minor son libres de implementar las soluciones en el lenguaje de programación *C++*, *Java* o *Python*, en este caso pueden tomar el código proporcionado como una base para empezar a realizar los ejercicios. **Para el caso de entregas en Python:** En la implementación de **LinkedList** deben crear una clase **nodo** auxiliar, mientras que para **ArrayList** se debe utilizar **array**<sup>34</sup> de *Python*.

## Normas de entrega

Antes del siguiente laboratorio, se deben enviar todos los ejercicios resueltos a los ayudantes mediante la plataforma CANVAS.

Se debe entregar un archivo comprimido que contenga:

- Archivo PDF con el nombre completo, número de matrícula, las respuestas a las preguntas que correspondan.
- Todos los ficheros del código fuente.
- **IMPORTANTE:** Los archivos debe llamarse *apellido1\_nombre\_04.formato*

---

<sup>1</sup><https://www.geeksforgeeks.org/new-and-delete-operators-in-cpp-for-dynamic-memory/>

<sup>2</sup><https://www.geeksforgeeks.org/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/>

<sup>3</sup><https://www.geeksforgeeks.org/python-arrays/>

<sup>4</sup><https://docs.python.org/3/library/array.html>