



Laboratorio de Estructuras de Datos

Práctica 12. Listas enlazadas, arreglos dinámicos

Unidad Temática: 2. Listas enlazadas

📖 👤 Profesor: Dr. Aldonso Becerra Sánchez

Índice

1	Objetivo de la tarea	1
2	Tiempo aproximado de realización	1
3	Fecha de entrega	1
4	Fecha de entrega con extensión y penalización	1
5	Introducción	1
6	Actividades a realizar	1
6.1	Actividad Inicial	1
6.2	Actividad 1	1
6.3	Actividad 2	1
6.4	Actividad 3	1
6.5	Actividad 4	1
6.6	Actividad 5	2
6.7	Actividad 6	2
6.8	Actividad 7	2
6.9	Actividad 8	2
7	Contáctame	2
	References	2

1. Objetivo de la tarea

Profundizar con las operaciones de listas dinámicas.

2. Tiempo aproximado de realización

🕒 5 horas.

3. Fecha de entrega

📅 4 de octubre de 2024.

4. Fecha de entrega con extensión y penalización

📅 5 de octubre de 2024.

5. Introducción

La memoria dinámica es un elemento importante en el manejo de información abundante donde no se sabe de antemano cuantos datos son los requeridos, por tanto solventa las limitaciones de la memoria estática. Las listas enlazadas permiten la manipulación de la memoria dinámica a través de la liga de nodos sucesivos [1], [2], [3].

6. Actividades a realizar

6.1. Actividad Inicial

Lea primero toda la práctica ⚠️. No inicie a programar sin leer todo cuidadosamente primero. Recuerde que debe generar el reporte en formato IDC con todos sus componentes.

6.2. Actividad 1

Primero genere la **Introducción** 📄.

6.3. Actividad 2

Información importante

Esta actividad debe entrar en la parte de *Desarrollo* 📄.

Defina la funcionalidad de la clase Pila utilizando arreglos dinámicos. Invoque métodos existentes.

6.4. Actividad 3

Información importante

Esta actividad debe entrar en la parte de *Desarrollo* 📄.

Implemente la funcionalidad de la clase Cola simple utilizando arreglos dinámicos. Invoque métodos existentes.

6.5. Actividad 4

Información importante

Esta actividad debe entrar en la parte de *Desarrollo* 📄.

Implemente el código de listas enlazadas: **ArregloListaInfoDinamica** (la lista no debe tener atributos que lleven el control numérico de cuántos elementos existen actualmente en la lista dinámica:

1. Imprimir en orden inverso los elementos: **void imprimirOI()**. Use pilas para resolverlo.


2. Realizar el método que permita encontrar un elemento en una lista encadenada. El método debe regresar el contenido del info encontrado: **ArregloListaInfoDinamica buscarObjetos(Object valor)**. Lo interesante de este método es que se debe regresar todas las ocurrencias del objeto encontrado.

3. Hacer un método que guarde todos los elementos de una lista dinámica en un arreglo, el cual debe regresar como valor de retorno: **ArregloListaInfoEstatica aArregloEstatico()**.
4. Hacer un método que guarde todos los elementos de una lista a un arreglo siempre y cuando no sean iguales a los que contiene el arreglo pasado como argumento: **ArregloListaInfoEstatica aArregloEstatico(ArregloListaInfoEstatica elementosA-Descartar)**.
5. Hacer un método que guarde los elementos de una lista en una matriz 2d. A este método se le pasarán el número de renglones y columnas de la matriz resultante. En caso que no se ajusten de renglones o columnas con la cantidad de elementos de la lista, se deben rellenar con null: **ArregloListaInfoEstatica2 aMatriz2(int filas, int columnas)**.
6. Hacer un método que agregue al final de la lista los elementos pasados como argumentos en un arreglo estático o un arreglo dinámico (listadatos2). **boolean agregarLista(ArregloListaDatos listaDatos2)**.
7. Hacer un método que devuelva una copia de la lista ligada. La primera lista debe ser independiente de la copia: **ArregloLista-Datos copiar()**.
8. Hacer un método que agregue los elementos de una matriz 2d pasada como argumento al final de una lista. Los elementos irán agregando renglón por renglón o columna por columna, según se defina a través de una enumerado (COLUMNA; RENGLON): **boolean agregarMatriz2(ArregloListaInfoEstatica2 tabla, TipoTabla enumTipoTabla)**.
9. Hacer el método que vacíe una lista ligada. **void resetear()**.
10. Hacer un método que rellene una lista con valores iguales indicados por argumento. **void llenar(Object valor, int cantidad)**.
11. Hacer un método que cuente elementos en una lista con valores iguales indicados por el argumento. **int contar(Object valor)**.
12. Hacer un método que invierta el orden de una lista. **void invertir()**.
13. Hacer un método que cambie un elemento viejo por uno nuevo dado un número de ocurrencias. **boolean modificar(Object valorViejo, Object valorNuevo, int cuantasVeces)**.
14. Hacer un método que cambie un elemento viejo por uno nuevo dada una posición de búsqueda: **boolean modificar(int indice, Object valor)**.
15. Hacer un método que devuelva el elemento que se encuentra en una posición dada como argumento: **Object dato(int indice)**.
16. Hacer un método que indique si una lista es igualita a otra lista. Elemento por elemento; en la misma posición deben ser iguales. **boolean esIdentico(ArregloListaDatos listaDatos2)**.
17. **public boolean cambiarTamano(int maximo)**: redimensiona el tamaño de la lista al nuevo tamaño indicado por máximo. Si el tamaño es menor, los elementos sobrantes deben ser eliminados. Si el tamaño es mayor, los datos anteriores deben conservarse, y los nuevos espacios deben llenarse con null.
18. **public Object eliminar(int indice)**: elimina un elemento de la lista en una posición específica, regresando el elemento eliminado.
19. **public ArregloListaInfoDinamica eliminarElementos(ArregloListaDatos valor)**: elimina todos los elementos de la lista dado un valor específico, regresando los elementos eliminados.

6.6. Actividad 5

Realice la sección de Código agregado  (diagrama de clases UML).





6.7. Actividad 6

Realice la sección de Pre-evaluación  (use los lineamientos establecidos).

6.8. Actividad 7

Finalmente haga las **Conclusiones** .


6.9. Actividad 8


Subir los entregables (pdf  y zip  con código )  a Moodle.

7. Contáctame

Puedes contactarme a través de los siguientes medios.

 <https://moodle.ingsoftware.uaz.edu.mx/>

 a7donso@gmail.com

 Cubículo

 Salón CC2-IS

Referencias

- [1] O. Cairo y S. Guardati, *Estructura de datos*. McGraw-Hill.
- [2] L. Joyanes Aguilar, *Fundamentos de programación, algoritmos u estructura de datos*. McGraw-Hill.
- [3] M. A. Weiss, *Estructura de datos en Java*. Addison Wesley.