



Laboratorio de Estructuras de Datos

Práctica 2. Arreglos unidimensionales

Unidad Temática: 1. Introducción a las estructuras de datos y estructuras fundamentales

📅 👤 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	2
6.5	Actividad 4	2
6.6	Actividad 5	2
6.7	Actividad 6	2
6.8	Actividad 7	2
7	Contáctame	2
	References	2

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* 📄.

Complete el TDA llamado “ArregloListaInfoEstatica”, el cual debe tener atributos y métodos necesarios para manipularlo en inserción, búsqueda, modificación y eliminación de forma desordenada. Para completar muchos de estos métodos necesitará crear los métodos `int tamanio()`, que regresará el tamaño máximo del arreglo, o el método `int numDatos()`, que regresa la cantidad de elementos existentes en el arreglo. Estos métodos deben ir desde la interfaz `VectorArregloDatos` (por ser memoria estática).

Se le pide además agregar:

- **Declarado en interface ArregloListaDatos.** `public boolean esIdentico(Object arreglo2)`: indica si el arreglo actual es igual a la `arreglo2`. La igualdad se determina por el contenido de cada elemento en la posición correspondiente al mismo índice. Debe validar que `arreglo2` sea un `ArregloListaInfoEstatica`.
- **Declarado en interface VectorArregloDatos.** `public Object dato(int posicion)`: regresa el objeto de la posición “posicion”. Recuerde validar que la posición que se pasa como argumento debe estar en un rango válido, en caso contrario regrese `null`.
- **Declarado en interface ArregloListaDatos.** `public boolean modificar(Object valorViejo, Object valorNuevo, int cuantasVeces)`: modifica el elemento viejo por el elemento nuevo haciendo una búsqueda y modificación del número de veces de ocurrencias del elemento encontrado indicado por `cuantasVeces`. Regrese verdadero si hizo alguna modificación. No olvide las validaciones de rangos de índices.
- **Declarado en interface VectorArregloDatos.** `public boolean cambiar(int posicion, Object valor)`: modifica el elemento de la posición indicada por “posicion”. “valor” es el nuevo elemento a colocar. Realice el proceso de validación para que no permita

1	1. Objetivo de la tarea	1
2	Reacción de TDA arreglo (ArregloListaInfoEstatica) para su posterior uso en aplicaciones comunes.	1
3		
4	2. Tiempo aproximado de realización	1
5	📅 ⌚ 5 horas.	1
6	3. Fecha de entrega	1
7	📅 16 agosto de 2024.	1
8	4. Fecha de entrega con extensión y penalización	1
9	📅 17 agosto de 2024.	1
10	5. Introducción	1
11	Existe diversidad de usos que se les puede dar a los arreglos. Desde este punto de vista, los arreglos propician que muchos planteamientos puedan tener una solución sencilla si se llevan a cabo con la ayuda de ellos. [1], [2], [3].	1
12		
13		
14		

cambiar nodos inexistentes en “posicion”. Regrese verdadero si sí pudo o falso si no se pudo.

- **Declarado en interface VectorArregloDatos.** public boolean modificar(ArregloListaInfoEstatica posicionesBusqueda, ArregloListaInfoEstatica valoresNuevos): modifica el elemento del arreglo actual en las posiciones de “posicionesBusqueda” por el contenido de “valoresNuevos”. De esta forma se tendrán nuevos valores en cada una de las posiciones del arreglo viejo por cada una de las posiciones del arreglo nuevo tomando como base los “posicionesBusqueda”. Regrese verdadero si sí pudo o falso si no se pudo o no debido a las dimensiones o índices inválidos, por ejemplo.
- **Declarado en interface ArregloListaDatos.** public ArregloListaInfoEstatica buscarValores(Object valor): busca dentro de un arreglo los elementos indicados por valor, si existen muchos elementos o en su caso uno; debe guardar en un arreglo (que devolverá) las posiciones de todas y cada una de las ocurrencias del elemento buscado.
- **Declarado en interface VectorArregloDatos.** public Object quitar(int posicion): elimina un elemento del arreglo en una posición específica, regresando el elemento eliminado.
- **Declarado en interface ArregloListaDatos.** public Object quitar(): regresa el objeto de la última posición del arreglo.
- **Declarado en interface ArregloListaDatos.** public void limpiar(): vacía el contenido de un arreglo.
- **Declarado en interface ArregloListaDatos.** public boolean agregarArreglo(Object arreglo2): agrega al final del arreglo actual (en este caso el TDA ArregloListaInfoEstatica) el contenido de la arreglo2 (en este caso otro arreglo). Debe validar que arreglo2 sea un ArregloListaInfoEstatica.
- **Declarado en interface ArregloListaDatos.** public void voltear(): invierte el orden de los elementos de un arreglo.
- **Declarado en interface ArregloListaDatos.** public int cuantificar(Object valor): contar cuántos elementos hay en el arreglo con el contenido igual a valor especificado como argumento.
- **Declarado en interface ArregloListaDatos.** public boolean quitarArreglo(Object arreglo2): elimina cada elemento de arreglo2 que se encuentre en el arreglo actual (en este caso el TDA ArregloListaInfoEstatica). Debe validar que arreglo2 sea una ArregloListaInfoEstatica.
- **Declarado en interface ArregloListaDatos.** public void llenar(Object valor, int numElementos): rellena todos los elementos indicados por “numElementos” del arreglo con el “valor” indicado. Validar que cantidad no se pase del tamaño de la estructura, en caso que sea estática; sino sólo rellenar hasta la capacidad máxima.
- **Declarado en interface ArregloListaDatos.** public ArregloListaDatos copiar(): regresa una copia del arreglo actual.
- **Declarado en interface ArregloListaDatos.** public ArregloListaDatos subArreglo(int posicionInicial, int posicionFinal): regresa un arreglo con los elementos indicados con las posiciones inicial y final del arreglo actual. Valide los rangos.
- **Declarado en interface VectorArregloDatos.** public boolean cambiarTamano(int max): redimensiona el tamaño del arreglo 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 estar vacíos.


6.4. Actividad 3

Pruebe el funcionamiento del programa de la actividad 2 con todo y sus capturas de pantalla.

6.5. Actividad 4

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





6.6. Actividad 5

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

6.7. Actividad 6

Finalmente haga las Conclusiones .


6.8. Actividad 7


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.