FCEFQyN, Universidad Nacional de Río Cuarto
Asignatura: Estructuras de Datos y Algoritmos - Algoritmos I
Segundo Cuatrimestre de 2020

Práctica 2

Parte de esta práctica cuenta con esquemas de programación que pueden accederse a partir del repositorio: https://github.com/EstructurasUNRC/practicas-algoritmos.git
Importante: El código provisto como parte de la solución a los ejercicios de la práctica, debería estar documentado apropiadamente, es decir, especificar los métodos mediante pre- y post-condiciones (como comentarios en el código).

Ej. 0. Lea las secciones 3.1 a la 3.5 del capítulo 3 de Data Structures and Problem Solving Using Java (Mark Allen Weiss 4th Edition).

Ej. 1. Acceder al directorio `ejercicios-practica-2/ejercicio1/` del repositorio de la materia, que incluye la implementación de una clase Libro, y una clase Main que construye algunos libros y los muestra por pantalla. Crear algunos libros adicionales y mostrar su contenido.

Ej. 2. Acceder al directorio `ejercicios-practica-2/ejercicio2/` del repositorio de la materia,, que agrega un método equals para comparar libros por igualdad. Estudie el método equals e interprete los resultados de ejecutar el método main de la clase Main.

Ej. 3. Acceder al directorio `ejercicios-practica-2/ejercicio3/` del repositorio de la materia,y complete la implementación del catálogo de libros. Este utiliza un arreglo para almacenar los libros, y provee métodos para agregar un libro al final del arreglo y buscar un libro por título. Su tarea es implementar estos dos últimos métodos.

Ej. 4. Implemente en Java el TAD Pila genérico.

a. Implemente un algoritmo que tome como argumento una secuencia de números y los imprima en orden inverso. Ayuda: Utilice una pila.
b. Agregue una implementación de Pilas, diferente a la definida al comienzo de este ejercicio, es decir, si implementó una pila con arreglos cree una nueva pila con listas enlazadas y viceversa. Para las pilas con arreglos lance una excepción en caso de que la pila se llene.

c. Cambie la implementación de pilas utilizada en el inciso a. modificando lo menos posible el código.

Ej. 5. Compare la implementación del TAD Pila definido en el Ej.4 con la implementación de Pila definido en el lenguaje C, en la práctica de repaso.

Ej. 6. Implemente un algoritmo que tome como argumentos una secuencia de paréntesis, corchetes y llaves y determine si la secuencia está balanceada. Usar una pila. Por ejemplo, su algoritmo deberá retornar verdadero para [()][()()]() y falso para [(]).

Ej. 7. Modifique la implementación de pilas con arreglos para que cambie de tamaño dinámicamente, de manera que nunca se llene (hasta agotar la memoria).

Ej. 8. Acceder al directorio `ejercicios-practica-2/queue/` del repositorio de la materia, que incluye clases que Implementan el TAD Cola (Queue.java) y realice los siguientes ejercicios:

a. Completar la implementación del método dequeue de la clase ListQueue.java.
b. Utilizando alguna de las implementaciones del TAD Cola, defina un programa que decida si una cadena, ingresada por línea de comandos, es un palíndromo o no.
c. Resuelva el mismo problema planteado en el inciso b, pero utilizando una implementación diferente del TAD Queue a la que usó para resolverlo en el inciso anterior y compare los resultados.
d. Defina en el lenguaje C++ una implementación del TAD Cola y luego compare las dos implementaciones del TAD.

Ej. 9. Escribir una clase para el manejo de números racionales. Sus atributos son dos variables de tipo long. Puede suponer que el denominador siempre es positivo. La clase debe proveer funcionalidades para sumar y restar racionales, y un método equals para comparar dos racionales por igualdad. Implementar también un método toString que convierta un racional a una cadena. Hacer un método main que cree varios racionales y les aplique diversas operaciones, mostrando sus resultados por pantalla.

Ej. 10. Incorporar a la clase Lista, las siguientes operaciones:

- Agregar los elementos de la lista a otra lista.
- Definir igualdad de listas.

• Crear una copia de la lista (sin copiar los elementos que la componen).

Ej. 11. Acceda al directorio `ejercicios-practica-2/ejercicio11/` del repositorio de la materia. Este contiene un método principal que toma el nombre de un archivo de texto y un número n, e imprime el archivo por pantalla. Modifique el algoritmo para que sólo imprima las últimas ***n*** líneas de los archivos. ¿Qué estructura de datos debería utilizar?

Ej. 12. (Opcional) Implemente una lista que posea una operación de concatenación de tiempo constante (sin usar ciclos). Ayuda: use una lista circular, manteniendo un puntero al último elemento de la lista.

Ej. 13.

a. El directorio `ejercicios-practica-2/ejercicio13/` del repositorio de la materia, contiene una clase LeerCadenas, para leer cadenas ingresadas por consola por el usuario. Modifique este algoritmo para implementar una lista de compras. El usuario debe ingresar los nombres de los artículos de a uno por vez. El programa debe almacenar los artículos e imprimirlos en el orden original al finalizar la confección de la lista. Utilizar una lista simplemente encadenada genérica.

b. Modifique el programa anterior para que el usuario pueda ingresar la cantidad de artículos a comprar por cada ítem en la lista.

Ej. 14. Provea una implementación de vectores de números reales en JAVA (ayuda: utilice arreglos). Las operaciones definidas sobre los vectores son: multiplicación por un escalar, suma y multiplicación (producto punto) de vectores del mismo tamaño. También deben poder compararse por igualdad. Cree algunos vectores y utilice sus operaciones.