DESARROLLO DE ALGORITMOS - 2º cuatrimestre 2022
Trabajo Práctico de Arreglos Unidimensionales

IMPORTANTE: Diseñe algoritmos aplicando conceptos de modularización. Lea detenidamente el enunciado y trate de pensar cuales son los módulos que debe definir. En cada algoritmo no olvide especificar un nombre, definir comentarios apropiadamente y especificar el tipo de todas las variables utilizadas. Cada vez que diseñe un módulo tenga en cuenta en primera instancia su propósito, sus parámetros formales y valor de retorno. En cada invocación a un módulo, revise cuidadosamente los parámetros actuales, el orden y la compatibilidad de tipos entre parámetros actuales y formales.

Para los siguientes ejercicios: Diseñar los algoritmos, realizar traza e implementar en Java

1. Diseñar un algoritmo en pseudocódigo que:
    a. Cargue un arreglo de caracteres.
    b. Permita al usuario elegir si lo quiere ver en el orden ingresado o invertido (Modularice apropiadamente)

2. Diseñar un algoritmo en pseudocódigo que lea un valor entero (N) y genere un arreglo con los **10 primeros múltiplos** del mismo.

Por ejemplo para N=7 deberá guardar en el arreglo: 7 14 21 28 35 42 49 56 63 70

3. Diseñar un algoritmo en pseudocódigo que dado un valor entero N y un arreglo de enteros, reemplace los valores en las **posiciones pares** del arreglo por el valor **N** y muestre el arreglo resultante.

4. Diseñar un algoritmo en pseudocódigo que permita encontrar **el valor más grande y el más pequeño** almacenado en un arreglo de números enteros. Asuma que el arreglo ingresa por parámetro al módulo.

5. Diseñar un algoritmo en pseudocódigo que calcule el **promedio de los valores** almacenados en un arreglo de números. Asuma que el arreglo ingresa por parámetro al módulo.

6. Diseñar un algoritmo en pseudocódigo que permita almacenar **letras** en un arreglo, cuya dimensión **máxima es de 100 posiciones**. El algoritmo debe verificar que el caracter leído sea una letra antes de guardarlo en el arreglo. **Puede ocurrir que el usuario no quiera ingresar los 100 elementos.** Al finalizar la carga el algoritmo debe mostrar por pantalla la cantidad de letras guardadas.

7. Diseñar un algoritmo en pseudocódigo que permita:
    a. Leer palabras y almacenarlas en un arreglo de string.
    b. Generar una cadena con las palabras almacenadas en el arreglo separándolas por un espacio en blanco
    c. Generar otra cadena con las palabras almacenadas en el arreglo en orden inverso separándolas por un guión ('-')
    d. Mostrar ambas cadenas por pantalla.

8. Diseñar un algoritmo en pseudocódigo que busque **la palabra más larga** almacenada en un arreglo de String (cada posición guarda exactamente 1 palabra).

9. Diseñar **dos módulos** en pseudocódigo que dado un **arreglo de caracteres y un caracter**:

a. **Verifique si el caracter ingresado se encuentra en el arreglo**. ¿Puede optimizar el algoritmo?

b. **Cuente cuántas veces aparece el caracter** en el arreglo. ¿Puede optimizar el algoritmo?

Implementar el algoritmo llamador que invoque a los módulos.

10. Diseñar un algoritmo en pseudocódigo que dado un arreglo cargado con valores enteros genere otro arreglo con los valores invertidos.

Por ejemplo si el arreglo contiene: 12 4 8 22 5, el nuevo arreglo será 5 22 8 4 12

11. Diseñar un algoritmo en pseudocódigo que **cargue dos arreglos de números** y luego **verifique si son iguales o no**. Para ello se debe implementar un módulo que realice la verificación.

12. Diseñar un algoritmo en pseudocódigo **que cargue un arreglo de caracteres** y luego **realice la copia** del mismo en otro de igual tamaño (modularice).

13. Diseñar un algoritmo en pseudocódigo **que cargue un arreglo de caracteres** y luego **genere otro** que contenga **solo las vocales** que se encuentran en el arreglo original.

14. Diseñar un algoritmo en pseudocódigo que **cargue un arreglo de String** y luego genere dos nuevos arreglos, uno conteniendo las cadenas que estaban en **las posiciones pares** y otro conteniendo los caracteres que estaban **en las posiciones impares.** Modularice.

15. Problema Número de DNI. El documento de identidad (DNI) en España, consta de 8 cifras y de una letra. La letra del DNI se obtiene siguiendo los pasos a continuación:

1) Calcula el resto de dividir el número del DNI entre 23
2) El número obtenido estará entre 0 y 22, selecciona la letra asociada al valor obtenido utilizando la siguiente tabla:

0	1	2	3	4	5	6	7	8	9	10	11
T	R	W	A	G	M	Y	F	P	D	X	B

12	13	14	15	16	17	18	19	20	21	22	
N	J	Z	S	Q	V	H	L	C	K	E	

Por ejemplo, si el número del DNI es 31415927 y el resto de dividir por 23 es 20, la letra que le corresponde según la tabla es la “C”
Diseñar un algoritmo que solicite un numero de 8 cifras y devuelva el número de DNI correspondiente.

16. Dado un arreglo que almacena cadenas de caracteres se desea verificar que las mismas cumplan con las siguientes condiciones: tengan una longitud mínima de 5 caracteres y que contenga solo letras. En caso de que la cadena no cumpla la condición debe ser eliminada del arreglo y la cadena que está en la siguiente posición debe ocupar su lugar. Imprima por pantalla el arreglo resultante.

17. Dado un arreglo que almacena las notas correspondientes a un alumno, las cuales son números reales, se desea verificar si el alumno aprobó el cuatrimestre. La condición para aprobar es tener todas las notas con valores mayores o iguales a 6.
Se debe implementar un algoritmo que cargue el arreglo con 10 notas y verifique si el alumno aprobó o no el cuatrimestre.

18. Implementar un algoritmo que utilice dos arreglos, uno que almacena nombres de personas empleadas en una empresa y otro que almacena los sueldos de las mismas, y sabiendo que ambos arreglos se corresponden por posición, presentar un menú de opciones para realizar algunas de las siguientes acciones:

a. Buscar la persona que tiene mayor sueldo, mostrar su nombre y el sueldo.
b. Listar todas las personas que cobran exactamente un valor X (leído por teclado).
c. Aumentar en un 10% los sueldos que sean inferiores a $10000.
d. Buscar una persona y si se encuentra mostrar su sueldo.