Arreglos

Programación I

Objetivos del tema

- Desarrollar aplicaciones con uso de arreglos de una dimensión
- Resolver problemas aplicando diseño descendente

Arreglos

- El arreglo es una estructura de datos que se representa mediante una colección de datos de un mismo tipo.
- Para definir un arreglo se debe indicar: el tipo de todos sus datos, la cantidad de datos que va tener y un nombre de variable que representa a todo el arreglo.
- Una vez definido el tamaño o la cantidad de datos del arreglo se considera que no cambia (aunque algunos lenguajes lo permiten no lo vamos a usar de esa manera).
- A cada dato del arreglo se puede acceder en forma directa o aleatoria a través del nombre del arreglo y la posición de donde se ubica el dato dentro del arreglo.
- El uso de la variable arreglo junto con una posición se considera que es una variable del mismo tipo de datos con que fue creado el arreglo.

Declaración de arreglo

- Declarar un arreglo como colección de datos de un tipo tiene la forma **tipo_dato [] nombre_arreglo;** por ejemplo:

```
int [] arrentero;
//arrentero es un arreglo de enteros
```

- En la declaración anterior arrenteros no tiene datos, solo es una referencia de donde estarán los datos. Para hacer el espacio donde estarán los datos se realiza **nombre_arreglo = new tipo_dato[CANTIDAD];** por ejemplo:

```
arrentero = new int [MAX];
//Suponiendo que MAX es una constante entera > 0, la anterior
//sentencia asigno a arrenteros espacio para contener MAX enteros.
```

- La asignación de espacio se puede hacer de manera explicita a partir de valores **nombre_arreglo = {valor1, valor2,…,valorN};** por ejemplo:

```
arrentero = {1, 3, 5}; arrenteros tiene 3 valores enteros: 1, 3 y 5.
```

- La asignación de espacio explícita solo se utiliza cuando se pide hacerlo de esa manera (se utiliza muy poco).

Acceso al arreglo

- Una vez que el arreglo tiene espacio asignado el paso siguiente es inicializarlo con valores o cargarlo desde teclado. Para acceder a cada espacio del arreglo se utiliza un valor de posición o variable con dicho valor, que va desde **0** a la **CANTIDAD-1** de espacio asignado que tiene. Por ejemplo:

```
arrentero[0] = 1;
//se accede a la posición 0 del arreglo y se le asigna un 1
pos = 2; //suponiendo que pos es un entero
arrentero[pos] = 5;
//se accede a la posición pos del arreglo y se le asigna un 5
//pos tiene que tener un valor entre 0 y CANTIDAD-1 de elementos del
//arreglo, sino dará ERROR
```

- El acceso a una posición mediante un valor no se recomienda, siempre se utiliza una variable entera cuyo valor va de **0** a la **CANTIDAD-1** de espacio que tiene el arreglo.

Ejemplo 1

```
/*Hacer un programa que cargue en un arreglo de enteros 5 valores desde teclado y lo imprima.
ESTE EJEMPLO ESTA HECHO SIN METODOS, SOLO PARA EXPLICAR COMO FUNCIONA (MAS ADELANTE SE HACE CON
METODOS)
*/
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Clase_7_Ejemplo_1 {
   public static int MAX = 5;
   public static void main (String [] args) {
       int [] arrenteros = new int [MAX];
       BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));
       try{
          for (int pos = 0; pos < MAX; pos++){
                   System.out.println("Ingrese un entero: ");
                   /*arrenteros es un arreglo, arrenteros[pos] es como una
                   variable de tipo entero a la que se le asigna un valor
                   */
                   arrenteros[pos] = Integer.valueOf(entrada.readLine());
          }
          for (int pos = 0; pos < MAX; pos++){
                   System.out.println("arrenteros["+pos+"] -> "+arrenteros[pos]);
       }
       }
       catch(Exception exc){
          System.out.println(exc);
       }
   }
}
```

Cuando se ejecuta:

```
Ingrese un entero:
1
Ingrese un entero:
3
Ingrese un entero:
2
Ingrese un entero:
5
Ingrese un entero:
4
arrenteros[0] -> 1
arrenteros[1] -> 3
arrenteros[2] -> 2
arrenteros[3] -> 5
arrenteros[4] -> 4
```

Pasaje de parámetros

Pasaje de parámetro por valor

- Cuando se invoca a un método se pasa por parámetro una copia de los valores respetando el orden y tipo según como esta declarado el método invocado.
- Los cambios de la variable parámetro que figura en la declaración **NO impactan** en la variable que figura en la invocación si son de tipos primitivos.

Pasaje de parámetro por referencia

- Cuando se invoca a un método, la variable que figura en el parámetro de la invocación se asocia a la variable que figura en la declaración (como si fuera una sola con dos nombres).
- Los cambios de la variable parámetro que figura en la declaración **impactan** en la variable que figura en la invocación.

Pasaje de parámetros en Java

- **En Java todos los parámetros se pasan por valor.**
- **Cuando el parámetro es de tipo primitivo**, se pasa una copia del valor de la variable que figura en la invocación y se lo asigna a la variable parámetro que figura en la declaración. **El método no puede modificar la variable que figura en la invocación.**
- **Cuando el parámetro no es de tipo primitivo (por ejemplo, un arreglo),** también se pasa una copia del valor de la variable. **Pero este valor no es el valor de un tipo primitivo, es un dirección de memoria**. Esa copia se lo asigna a la variable parámetro que figura en la declaración. El valor original y la copia (direcciones de memoria) pueden acceder a la misma colección de datos que incluye. Así, **un método puede modificar el contenido de la variable de tipo no primitivo que figura en la invocación.**

Ejemplo 1

```
/*Hacer un programa que cargue en un arreglo de enteros 5 valores desde teclado y lo imprima.
*/
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Clase_7_Ejemplo_1 {
   public static int MAX = 5;
   public static void main (String [] args) {
        int [] arrenteros = new int [MAX];
        cargar_arreglo_int(arrenteros);
        imprimir_arreglo_int(arrenteros);
   }
   public static void cargar_arreglo_int(int [] arr){
        BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));
        try{
            for (int pos = 0; pos < MAX; pos++){
                      System.out.println("Ingrese un entero: ");
                      //modifico arr[pos], entonces modifico arrenteros[pos]
                      arr[pos] = Integer.valueOf(entrada.readLine());
               }
        }
        catch(Exception exc){
            System.out.println(exc);
        }
    }
   public static void imprimir_arreglo_int(int [] arr){
        for (int pos = 0; pos < MAX; pos++){
            //solo accedo a arr[pos] y no lo modifico, entonces no modifico arrenteros[pos]
            System.out.println("nombre_arreglo["+pos+"]=>: "+arr[pos]);
        }
   }
}
```

arrenteros y arr tienen distintos valores (de memoria) pero que acceden a la misma colección de datos. Si modifico arr[una_posicion] dentro del método también modificará arrenteros[una_posicion]

Cuando se ejecuta:

Ingrese un entero:
1
Ingrese un entero:
3
Ingrese un entero:
2
Ingrese un entero:
5
Ingrese un entero:
4
arrenteros[0] -> 1
arrenteros[1] -> 3
arrenteros[2] -> 2
arrenteros[3] -> 5
arrenteros[4] -> 4

Carga de arreglos

- Con el fin de simplificar los ejercicios, vamos a suponer que siempre nos dan arreglos cargados con datos. Para hacer pruebas de como funciona un código pueden usar un método que cargue desde consola. O podrán optar por cargas aleatorias.
- A continuación se da un ejemplo de carga de arreglo de enteros desde teclado. En la pagina que sigue se dan ejemplos de carga aleatoria e impresión de arreglos de char, int y double. La idea es reutilizarlos pero no se van a pedir.

```
public static void cargar_arreglo_int(int [] arr){
  BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));
  try{
    for (int pos = 0; pos < MAX; pos++){
      System.out.println("Ingrese un entero: ");
      arr[pos] = Integer.valueOf(entrada.readLine());
    }
  }
  catch(Exception exc){
      System.out.println(exc);
   }
}
```

Ejemplo 2: carga aleatoria e impresión de arreglos de char, int y double

```
//PARA ALEATORIO O RANDOM INCORPORAR LA LIBRERIA QUE SIGUE
import java.util.Random;
public class Clase_7_Ejemplo_2 {
  public static final int MAX = 10;
  public static final int MAXVALOR = 10;
  public static final int MINVALOR = 1;
  public static void main(String[] args) {
    char [] arrchar;
    int [] arrint;
    double [] arrdouble;
    arrchar = new char[MAX];
    arrint = new int[MAX];
    arrdouble = new double[MAX];
    cargar_arreglo_aleatorio_char(arrchar);
    cargar_arreglo_aleatorio_int(arrint);
    cargar_arreglo_aleatorio_double(arrdouble);
    imprimir_arreglo_char(arrchar);
    imprimir_arreglo_int(arrint);
    imprimir_arreglo_double(arrdouble);
  }
  //carga de arreglo de char con valores de 'a' a la 'z'
  public static void cargar_arreglo_aleatorio_char(char [] arr){
    Random r = new Random();
    for (int pos = 0; pos < MAX; pos++){
     arr[pos]=(char)(r.nextInt(26) + 'a');
    }
  }
  //carga de arreglo de int con valores de MINVALOR a MAXVALOR
  public static void cargar_arreglo_aleatorio_int(int [] arr){
    Random r = new Random();
    for (int pos = 0; pos < MAX; pos++){
     arr[pos]=(r.nextInt(MAXVALOR-MINVALOR+1) + MINVALOR);
    }
  }
  //carga de arreglo de double con valores de MINVALOR a MAXVALOR
  public static void cargar_arreglo_aleatorio_double(double [] arr){
    Random r = new Random();
    for (int pos = 0; pos < MAX; pos++){
     arr[pos]=((MAXVALOR-MINVALOR+1)*r.nextDouble() + MINVALOR*1.0);
    }
  }
  //impresion de arreglo de char
  public static void imprimir_arreglo_char(char [] arr){
    for (int pos = 0; pos < MAX; pos++){
     System.out.println("nombre_arreglo["+pos+"]=>: "+arr[pos]);
    }
  }
  //impresion de arreglo de int
  public static void imprimir_arreglo_int(int [] arr){
    for (int pos = 0; pos < MAX; pos++){
     System.out.println("nombre_arreglo["+pos+"]=>: "+arr[pos]);
    }
  }
  //impresion de arreglo de double
  public static void imprimir_arreglo_double(double [] arr){
    for (int pos = 0; pos < MAX; pos++){
     System.out.println("nombre_arreglo["+pos+"]=>: "+arr[pos]);
    }
  }
}
```

Ejemplo 3

```
/*Hacer un programa que dado un arreglo de enteros de tamano 10 que se encuentra
  precargado, imprima por pantalla el promedio de la suma de sus valores.
*/
public class Clase_7_Ejemplo_3 {
  public static int MAX = 10;
  public static void main (String [] args) {
        int [] arrenteros = new int [MAX];
        int promedio;
        //cargar el arreglo con alguno de los metodos propuestos
        cargar_arreglo_int(arrenteros);
        promedio = promedio_arreglo(arrenteros);
        System.out.println("El promedio del arreglo es: "+promedio);
  }

  public static int promedio_arreglo(int [] arr){
        int suma = 0;
        for (int pos = 0; pos < MAX; pos++){
              suma+=arr[pos];
       }
        return (suma/MAX);
  }
}
```

Como trabajar con arreglos

- Una forma de encarar los ejercicios de arreglos es partir de un esquema en pseudocodigo como el que sigue (**cada uno puede optar por hacerlo o no, los pseudocodigos no se piden en la resolución**):

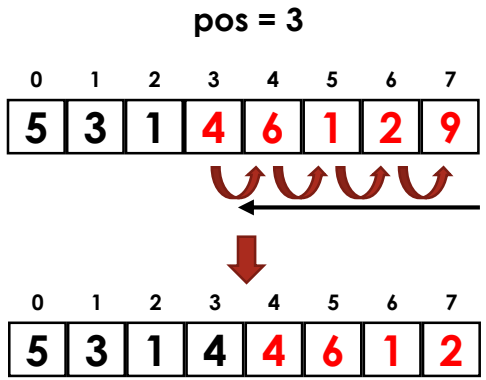
```
Programa{
   Definir constantes;
   Dentro de main(){
      Definir variables;
      Cargar arreglo/s;
      Realizar las tareas con arreglos;
      Imprimir arreglos/s o los resultados;
   }
}
```

Ejemplo 4

```
/*Hacer un programa que dado un arreglo de enteros de tamano 10 que se encuentra precargado, encuentre la posicion de un numero entero
   ingresado por el usuario. Si existe, muestre esa posicion por pantalla, o indique que no existe.
*/
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Clase_7_Ejemplo_4 {
   public static int MAX = 10;
   public static void main (String [] args) {
        int [] arrenteros = new int [MAX];
        int pos,numero;
        BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));
        try{
            cargar_arreglo_aleatorio_int(arrenteros);
            imprimir_arreglo_int(arrenteros);
            System.out.println("Ingrese un numero entero :");
            numero = Integer.valueOf(entrada.readLine());
            pos = obtener_pos_arreglo(arrenteros,numero);
            if (pos<MAX){
                        System.out.println(numero + " esta en " + pos);
            }
            else{
                        System.out.println("No existe " + numero);
            }
        }
        catch(Exception exc){
            System.out.println(exc);
        }
   }
   public static int obtener_pos_arreglo(int [] arr, int numero){
        int posicion = 0;
        while ((posicion < MAX) && (arr[posicion] != numero)){
            posicion++;
        }
        return posicion;
    }
}
```

Ejemplo 5

```
/*Hacer un programa que dado un arreglo de enteros de tamano 8 que se encuentra precargado, solicite al usuario una posicion y
   realice un corrimiento a derecha o hacia la mayor posicion del arreglo. Ademas imprima el arreglo antes y despues del
   corrimiento
*/
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Clase_7_Ejemplo_5 {
   public static int MAX = 8;
   public static void main(String[] args) {
    int [] arrenteros;
    arrenteros = new int[MAX];
    int pos;
    BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));
         try{
             cargar_arreglo_aleatorio_int(arrenteros);
             imprimir_arreglo_int(arrenteros);
             System.out.println("Ingrese un numero entero pos:");
             pos = Integer.valueOf(entrada.readLine());
             corrimiento_der(arrenteros,pos);
             imprimir_arreglo(arrenteros);
         }
         catch(Exception exc){
             System.out.println(exc);
          }
   }
   public static void corrimiento_der(int [] arrenteros, int pos){
         int indice = MAX-1;
         while (indice > pos){
             arrenteros[indice] = arrenteros[indice-1];
             indice--;
          }
     }
}
```



Práctico – primera parte

1. Hacer un programa que dado un arreglo de enteros de tamaño 10 que se encuentra precargado, invierta el orden del contenido (por ejemplo: el que está en 0 se intercambia con el que está en 9, el que está en 1 con el que está en 8…). Este intercambio no se debe realizar de manera explícita, hay que hacer un método que incluya una iteración de intercambio.

2. Hacer un programa que dado un arreglo de enteros de tamaño 10 que se encuentra precargado, obtenga la cantidad de números pares que tiene y la imprima.

3. Hacer un programa que dado un arreglo de enteros de tamaño 10 que se encuentra precargado, solicite al usuario una posición y realice un corrimiento a izquierda o hacia la menor posición del arreglo.

4. Hacer un programa que dado un arreglo de enteros de tamaño 10 que se encuentra precargado, solicite al usuario un numero entero y lo agregue al principio del arreglo (posición 0). Para ello tendrá que realizar un corrimiento a derecha (se pierde el último valor del arreglo) y colocar el numero en el arreglo en la posición indicada.

5. Hacer un programa que dado un arreglo de enteros de tamaño 10 que se encuentra precargado, solicite al usuario un numero entero y elimine la primer ocurrencia de numero (un número igual) en el arreglo si existe. Para ello tendrá que buscar la posición y si está, realizar un corrimiento a izquierda (queda una copia de la última posición del arreglo en la anteúltima posición).

6. Hacer un programa que dado un arreglo de enteros de tamaño 10 que se encuentra precargado, solicite al usuario un numero entero y elimine todas las ocurrencia de numero en el arreglo si existe. Mientras exista (en cada iteración tiene que buscar la posición dentro del arreglo) tendrá que usar la posición para realizar un corrimiento a izquierda (quedarán tantas copias de la última posición del arreglo como cantidad de ocurrencias del número).

Objetivos del tema

- Realizar operaciones de ordenamiento, eliminación, e inserción
- Incorporar el concepto de arreglo de secuencias y realizar operaciones sobre el mismo

Arreglos ordenados

- Cuando se menciona que una estructura está ordenada se refiere a una propiedad que posee la estructura respecto de sus datos.
- En un arreglo ordenado los datos están ordenados desde la posición 0 a CANTIDAD-1 por su valor.
- El orden puede ser ascendente/creciente o descendente/decreciente.
- Los métodos que requieren recorridos para hacer una consulta o una modificación tendrán que considerar si el arreglo está o no ordenado, ya que los recorridos son diferentes.

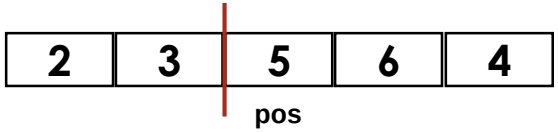
```
Desde la posición 0 a CANTIDAD-1
Arreglo de enteros desordenado =              2 1 6 5 7 3 2 5
Arreglo de enteros ordenado creciente =       1 2 2 3 5 5 6 7
Arreglo de enteros ordenado decreciente =     7 6 5 5 3 2 2 1
```

Ejemplo 6

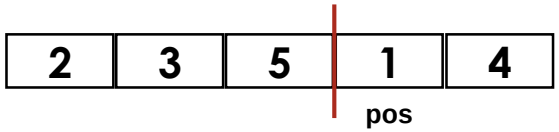
```
/*Hacer un programa que dado un arreglo de enteros ordenado creciente de tamano 10 que se encuentra precargado, encuentre la posicion
   donde se encuentra un numero entero ingresado por el usuario. Si existe, muestre esa posicion por pantalla, o indique que no existe.
*/
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Clase_7_Ejemplo_6 {
   public static int MAX = 10;
   public static void main (String [] args) {
    int [] arrenteros = new int [MAX];
    int pos,numero;
    BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));
    try{
         //CARGAR EL ARREGLO CON ALGUNO DE POR CONSOLA DE MANERA ORDENADA CRECIENTE
         System.out.println("Ingrese un numero entero :");
         numero = Integer.valueOf(entrada.readLine());
         pos = buscar_pos_arreglo_ord_crec(arrenteros,numero);
         if ((pos<MAX)&&(arrenteros[pos]==numero)){
              System.out.println("La posicion de "+numero+" es: "+pos);
         }
         else{
              System.out.println(numero + " no existe en el arreglo");
         }
    }
    catch(Exception exc){
         System.out.println(exc);
    }
    }
   //La posicion que retorna no significa que este ahí, es donde deberia estar
   public static int buscar_pos_arreglo_ord_crec(int[] arr,int numero) {
    int pos = 0;
    while ((pos<MAX)&&(arr[pos]<numero)){
         pos++;
    }
    return pos;
    }
}
```

Métodos de ordenamiento

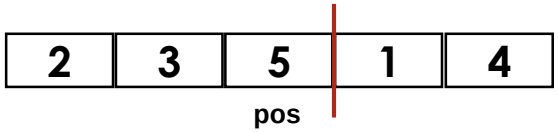
- Para ordenar un arreglo existen una serie de métodos bien documentados, con lo cual la idea es usar el algoritmo que propone cada método.
- De los métodos más usados vamos a mencionar 3 (dos ciclos o iteraciones para ordenar todo el arreglo):
  - Selección: en un ciclo iterativo, en un posición, lo de la izquierda está ordenado, y selecciona el más chico (segundo ciclo) de los que le siguen y lo intercambia. Luego avanza.



  - Inserción: en un ciclo iterativo, en una posición, lo que está a la izquierda está ordenado. Inserto ordenado (segundo ciclo) el elemento de esa posición a la izquierda. Luego avanza.



  - Burbujeo: en dos ciclos voy llevando siempre el más grande a la derecha intercambiando.



Ejemplo: selección, inserción y burbujeo

```
public class Clase_7_Ejemplo_ord {
   public static int MAX = 10;
   public static final int MAXVALOR = 10;
   public static final int MINVALOR = 1;
   public static void main(String[] args) {
   int [] arrint;
   arrint = new int[MAX];
   cargar_arreglo_aleatorio_int(arrint);
   imprimir_arreglo_int(arrint);
   //PROBAR LLAMAR LOS METODOS PARA ORDENAR (ARRINT);
   imprimir_arreglo_int(arrint);
   }


   public static void ordenar_arreglo_seleccion(int[]arr) {
   int pos_menor, tmp;
   for (int i = 0; i < MAX; i++) {
       pos_menor = i;
       for (int j = i + 1; j < MAX; j++){
           if (arr[j] < arr[pos_menor]) {
                     pos_menor = j;
           }
       }
       if (pos_menor != i){
           tmp = arr[i];
           arr[i] = arr[pos_menor];
           arr[pos_menor] = tmp;
       }
   }
   }
```

```
public static void ordenar_arreglo_insercion(int[]arr) {
int aux, j;
for (int i = 1; i < MAX; i++) {
     aux = arr[i];
     j = i - 1;
     while ((j >= 0) && (arr[j] > aux)){
         arr[j+1] = arr[j];
         j--;
     }
     arr[j+1] = aux;
}
}

public static void ordenar_arreglo_burbujeo(int[] arr){
int temp;
for(int i = 1;i < MAX;i++){
     for (int j = 0 ; j < MAX - 1; j++){
         if (arr[j] > arr[j+1]){
                   temp = arr[j];
                   arr[j] = arr[j+1];
                   arr[j+1] = temp;
         }
     }
}
}
```

Práctico – segunda parte

No hay que aplicar un método de ordenamiento cuando indica que el arreglo se encuentra ordenado.

7. Hacer un programa que dado un arreglo ordenado creciente de enteros de tamaño 10 que se encuentra precargado, solicite al usuario un numero entero y lo inserte en el arreglo manteniendo su orden. Para ello tendrá que realizar un corrimiento a derecha (se pierde el último valor del arreglo) y colocar el numero en el arreglo en la posición indicada.

8. Hacer un programa que dado un arreglo ordenado creciente de enteros de tamaño 10 que se encuentra precargado, solicite al usuario un numero entero y elimine la primer ocurrencia de numero (un número igual) en el arreglo si existe.

9. Hacer un programa que dado un arreglo de enteros de tamaño 10 que se encuentra precargado, solicite al usuario el ingreso de dos números enteros (posiciones del arreglo) y ordene de forma creciente el arreglo entre dos posiciones correspondientes a los números ingresados.

Arreglos de secuencias

- Los arreglos de secuencias es una forma de representar estructuras dentro de una estructura. Por ejemplo una oración con palabras separadas por espacios.
- Para ello se define una secuencia como un serie de valores continuos, y cada secuencia de separa de otra por separadores. Por ejemplo, una palabra es una secuencias de caracteres que no son espacio, y los separadores son los espacios. La oración sería el arreglo de caracteres.

	h	o	l	a				j	u	a	n			

- También se puede pensar en arreglos de números que representen estructuras dentro de una estructura. Por ejemplo, en un arreglo de enteros una secuencia de números representa los colores de una figura, que se separa de otra figura por separadores de un solo color blanco (255).

255	30	25	10	255	255	255	2	255	255	1	6	12	255	255

Arreglos de secuencias

- Trabajar con estas estructuras requiere de métodos que simplifiquen su recorrido, procesar las secuencias, incorporar secuencias, y eliminar secuencias.
- Para ello vamos a partir de una solución para cargar de forma aleatoria un arreglo de secuencias, que nos va permitir utilizarlo para probar nuestros ejercicios.
- Los métodos de carga de secuencias no se van a pedir en la resolución de ejercicios.
- A continuación se dejan ejemplos de códigos (carga e impresión) para reutilizar de arreglos de secuencias de caracteres letras minúsculas separadas por espacios, y de números enteros entre 1 y 9 separados por 0.

Ejemplo 7

```
import java.util.Random;
public class Clase_7_Ejemplo_7 {
   public static final int MAX = 40;
   public static final int MAXVALOR = 9;
   public static final int MINVALOR = 1;
   public static final double probabilidad_letra = 0.4;
   public static final double probabilidad_numero = 0.4;
   public static void main(String[] args) {
    char [] arrchar;
    int [] arrint;
    arrchar = new char[MAX];
    arrint = new int[MAX];
    cargar_arreglo_aleatorio_secuencias_char(arrchar);
    imprimir_arreglo_secuencias_char(arrchar);
    cargar_arreglo_aleatorio_secuencias_int(arrint);
    imprimir_arreglo_secuencias_int(arrint);
   }

   public static void cargar_arreglo_aleatorio_secuencias_char(char
   [] arr){
    Random r = new Random();
    arr[0] = ' ';
    arr[MAX-1] = ' ';
    for (int pos = 1; pos < MAX-1; pos++){
         if (r.nextDouble()>probabilidad_letra){
             arr[pos]=(char)(r.nextInt(26) + 'a');
         }
         else{
             arr[pos]=' ';

         }
    }
   }
```

```
   public static void imprimir_arreglo_secuencias_char(char [] arr){
    System.out.print("Arreglo de secuencias char\n|");
    for (int pos = 0; pos < MAX; pos++){
         System.out.print(arr[pos]+"|");
    }
    System.out.print("\n");
   }

   public static void cargar_arreglo_aleatorio_secuencias_int(int []
   arr){
    Random r = new Random();
    arr[0] = 0;
    arr[MAX-1] = 0;
    for (int pos = 1; pos < MAX-1; pos++){
         if (r.nextDouble()>probabilidad_numero){
             arr[pos]=(r.nextInt(MAXVALOR-MINVALOR+1) + MINVALOR);
         }
         else{
             arr[pos]=0;
         }
    }
   }

   public static void imprimir_arreglo_secuencias_int(int [] arr){
    System.out.print("Arreglo de secuencias int\n|");
    for (int pos = 0; pos < MAX; pos++){
         System.out.print(arr[pos]+"|");
    }
    System.out.print("\n");
   }
}
```

Práctico - tercera parte

Se tiene un arreglo de enteros de tamaño 20 de secuencias de números entre 1 y 9, separadas por 0. El arreglo esta precargado, y además empieza y termina con uno o más separadores 0. Considere para los siguientes ejercicios este tipo de arreglo.

10. Hacer un programa que dado el arreglo definido y precargado permita obtener a través de métodos la posición de inicio y la posición de fin de la secuencia ubicada a partir de una posición entera ingresada por el usuario. Finalmente, si existen imprima por pantalla ambas posiciones obtenidas.

Ejemplo 8

```
/*Hacer un programa que dado el arreglo definido y precargado, imprima lo que suma el contenido de cada secuencia.
 */
import java.util.Random;
public class Clase_7_Ejemplo_8 {
   public static final int MAX = 20;
   public static final int MAXVALOR = 9;
   public static final int MINVALOR = 1;
   public static final double probabilidad_numero = 0.4;
   public static void main(String[] args) {
         int[] arr;
         arr = new int[MAX];
         cargar_arreglo_aleatorio_secuencias_int(arr); //REUTILIZAMOS
         imprimir_arreglo_secuencias_int(arr); //REUTILIZAMOS
         imprimir_suma_cada_secuencia(arr);
   }
   public static void imprimir_suma_cada_secuencia(int[] arr){
         int inicio,fin,suma;
         inicio = 0;
         fin = -1;
         while ((inicio < MAX)){
              inicio = obtener_inicio_secuencia(arr,fin+1); //REUTILIZAMOS
              if (inicio < MAX){
                        fin = obtener_fin_secuencia(arr,inicio); //REUTILIZAMOS
                        suma = obtener_suma_secuencia(arr,inicio,fin);
                        System.out.println("La suma de la secuencia de "+inicio+" a "+fin+" es "+suma);
              }
         }
   }
   public static int obtener_suma_secuencia(int[] arr, int inicio, int fin){
         int suma = 0;
         while (inicio <= fin){
              suma+=arr[inicio];
              inicio++;
         }
         return suma;
   }
}
```

Práctico - tercera parte

11. Hacer un programa que dado el arreglo definido y precargado permita encontrar la posición de inicio y fin de la secuencia cuya suma de valores sea mayor.

12. Hacer un programa que dado el arreglo definido y precargado permita encontrar la posición de inicio y fin de la anteúltima secuencia (considerar comenzar a buscarla a partir de la ultima posición del arreglo).

13. Hacer un programa que dado el arreglo definido y precargado, y un número entero ingresado por el usuario, elimine las secuencias de tamaño igual al número ingresado.

14. Hacer un programa que dado el arreglo definido y precargado, y un número entero ingresado por el usuario, copie de forma continua las secuencias de tamaño igual al número ingresado en otro arreglo de iguales características e inicializado con 0. La copia en este último arreglo deben comenzar desde el principio del mismo.

15. Hacer un programa que dado el arreglo definido y precargado elimine del arreglo todas las ocurrencias de una secuencia patrón dada por otro arreglo de iguales características (solo tiene esa secuencia). Al eliminar se pierden los valores haciendo los corrimientos.

16. Hacer un programa que dado el arreglo definido y precargado elimine todas las secuencias que tienen orden descendente entre sus elementos.

Indexación de arreglos

- Es posible encontrarnos con problemas donde la información o los datos están distribuidos en más de un arreglo.
  - Por ejemplo, un arreglo A de enteros está desordenado. Un arreglo B tiene los índices de A de los valores pares al principio del arreglo. Hay un arreglo C que tiene los índices de A que permite recorrer A de forma ascendente.

A = | 1| 5| 1| 9| 3| 2| 4| 6|

B = | 5| 6| 7|-1|-1|-1|-1|-1|

C = | 0| 2| 5| 4| 6| 1| 7| 3|

- Este tipo de información sobre una estructura base (A) a la cual se la puede recorrer con distintos criterios (B o C) se lo asocia con la indexación.
- En este sentido la información de la estructura base (A) no se replica ni se modifica para tener otro orden u otra información sobre su contenido, sino que se generan índices (B y C) para acceder de la forma requerida.

Indexación de arreglos

- ¿Cómo se accede a la información desde la indexación?:
  - B y C tienen índices de A, si queremos por ejemplo imprimir los valores pares de A usando B haremos:

A = | 1| 5| 1| 9| 3| 2| 4| 6|

B = | 5| 6| 7|-1|-1|-1|-1|-1|

C = | 0| 2| 5| 4| 6| 1| 7| 3|

```
int i = 0
while ((i<MAX)&&(B[i]!=-1)){
        System.out.println(A[B[i]]);
        i++;
        //i es índice de B
        //B[i] es un índice de A
        //A[B[i]] es un valor de A
}
```

  - Si queremos por ejemplo comparar si efectivamente C mantiene un orden de A haremos:

```
int i = 0;
while (i<MAX-1)&&(A[C[i]]<A[C[i+1]]){
        i++;
        //i es índice de C, C[i] es un índice de A, A[C[i]] es un valor de A,
        //A[C[i]] se compara con el que le sigue en el arreglo A[C[i+1]]
}
```

Ejemplo 9

- Dado un arreglo DESORDENADO de números entre 1 y 9 de tamaño MAX que tiene los elementos sin orden, se pide lo siguiente:
  - a) Cargar un arreglo ORDENADO de tamaño MAX (inicializado con -1). ORDENADO tiene posiciones del arreglo DESORDENADO de forma tal que permite recorrer de forma ascendente y consecutiva los valores pares del arreglo DESORDENADO (no se pide ordenar ORDENADO). ORDENADO tiene valores -1 al final cuando DESORDENADO tiene valores impares.
  - b) Para valores (elemento y posición) ingresados por el usuario, si elemento esta entre 1 y 9, es impar, y posición es valida, insertar el elemento en DESORDENADO en la posición indicada y luego actualizar el arreglo ORDENADO sin acceder a DESORDENADO.

Link solución Clase_7_Ejemplo_9.java

Práctico - cuarta parte

17. Para el Ejemplo 9, hacer:
   - c) cargar los índices de los valores pares de DESORDENADO en un arreglo PARES de tamaño MAX sin importar el orden.
   - d) ordenar PARES de forma descendente según los valores pares de DESORDENADO utilizando un método de ordenamiento.

18. Se tiene un arreglo ORIGINAL precargado de caracteres letras minúsculas de tamaño MAX que está ordenado de forma ascendente. Además se tiene un arreglo ORDEN1 precargado de tamaño MAX. ORDEN1 tiene posiciones de ORIGINAL de forma tal que permita recorrer de forma ascendente y consecutiva los caracteres vocales de ORIGINAL. ORDEN1 tendrá valores -1 al final en el caso de que ORIGINAL tenga consonantes. Se pide:
   - Dada una posición ingresada por el usuario desde teclado, eliminar la letra de dicha posición en ORIGINAL, y actualizar el arreglo ORDEN1.
   - Imprimir los caracteres vocales siguiendo el orden establecido en ORIGINAL.

19. Se tiene un arreglo ORIGINAL precargado de secuencias de números de tamaño MAX con ceros entre secuencias, al principio y al final del arreglo. Además se tiene un arreglo ORDEN1 precargado de tamaño MAX. ORDEN1 tiene posiciones de inicio de secuencia de ORIGINAL que permite recorrer de forma ascendente y consecutiva las secuencias que suman una cantidad par (ORDEN1 tiene valores -1 al final de las posiciones válidas). Se pide:
   - Dada una posición válida ingresada por el usuario desde teclado, eliminar la secuencia en ORIGINAL con inicio en esa posición y actualizar el arreglo ORDEN1.

Práctico - cuarta parte

20. Para el Ejemplo 19, hacer:
   - cargar los índices de inicio de todas secuencias en el arreglo ORDEN2 de tamaño MAX, que está inicializado con -1.
   - Suponiendo que hay un método que permite ordenar de forma ascendente un arreglo de las características de ORDEN2 (el método ordena los índices de inicio de secuencia de según el tamaño de las secuencias a los que hacen referencia), generar un arreglo ORDEN3 de tamaño MAX de índices de las secuencias de ORIGINAL con el orden mencionado (deberá definir además los encabezados de el/los método/s prexistentes y hacer las llamadas cuando los utilice).

21. Suponer que se tiene un arreglo precargado de caracteres letras minúsculas de tamaño MAX. Se tiene además los siguientes métodos:
   - Un método que retorna cuantas repeticiones en el arreglo tiene un carácter ingresado como parámetro.
   - Un método que ordena un arreglo de forma descendente por cantidad de caracteres que se repiten, por ejemplo genera | 't' | 't' | 't' | 'a' | 'a' | 'r' | 'e' | .
   - Hacer un programa que:
     - dado un numero positivo ingresado el usuario imprima por pantalla los caracteres vocales con cantidad mayor a la cantidad ingresada sin realizar recorridos innecesarios. Por ejemplo si se ingresa un 2 imprime 'a' y no sigue recorriendo el arreglo.
     - mientras el usuario ingrese una vocal consonante minúscula muestre por pantalla la cantidad de repeticiones de dicha consonante.
     - deberá definir además los encabezados de el/los método/s prexistentes y hacer las llamadas cuando los utilice.