Arreglos y Matrices

Array

Un arreglo es una **estructura de datos lineal** que consiste en una **colección de elementos** del **mismo tipo** (y por ende del **mismo tamaño** en **bytes**), almacenados en **memoria contigua** e identificados cada uno con un **índice**.

Direcciones de memoria (en este ejemplo, son números de 4 bytes)

200	204	208	212	216	220	224	228	232	236	240	244
11	9	17	89	1	90	19	5	3	23	43	99
0	1	2	3	4	5	6	7	8	9	10	11

índices (offset)

$$dir\ i = dir\ 0 + i * bytes$$

Arrays en Java

```
int[] array1 = new int[4];
// Array de 4 elementos.
int[] array2 = {1, 2, 3, 4};
// Array de 4 elementos
inicializados.
```

En Java, los arrays **son objetos**

Para declarar un array, indicamos el tipo de dato, seguido de corchetes. []

Para instanciar un array, utilizamos la palabra reservada **new** seguido del tipo de dato y corchetes []. Dentro de los corchetes, se **debe especificar el tamaño**, el cual es **fijo durante toda la vida del objeto**.

Alternativamente, se puede instanciar el array con datos inicializados, usando una lista {}.

```java
int[] array2 = {1, 2, 3, 4};
array2[0] = 5;
System.out.println(array2[0]); // 5
System.out.println(array2.length); // 4
for (int i : array2) {
    System.out.println(i);
    // Admite ciclo for-each
}
```

A diferencia de otros lenguajes, por ser objetos, admiten de serie algunas funcionalidades básicas, como el acceso al tamaño (que es un atributo).

Acá se muestran los más básicos (asignación y acceso a un dato, tamaño, iteración mediante un iterador).

Java.util.Arrays

```java
import java.util.Arrays;

int[] array2 = {1, 2, 3, 4};

System.out.println(Arrays.toString(array2));
// [1, 2, 3, 4]
Arrays.sort(array2); // Ordenamiento
// etc...
```

La clase Arrays de Java tiene implementados una extensa cantidad de métodos útiles para el manejo de arrays.

Matrices en Java

```java
int[][] matriz = new int[4][4];

matriz[0][0] = 5;
System.out.println(matriz[0][0]);
```

Las matrices son, en definitiva, arrays de arrays (también llamadas arrays bidimensionales). Su declaración, la instalación y uso son similares a las de los arrays.

De la misma forma, uno puede crear arrays multidimensionales, por ejemplo: `new int[3][3][3];`

Pasaje de arrays / matrices por parámetro

```java
public static void funcion(int[] array) {
    int tamanio = array.length;
    // Hago algo con el array.
}

public static void funcion(int[][] matriz) {
    int tamanio = matriz.length;
    // Hago algo con la matriz.
}
```

De forma similar a otros parámetros, se indica en la firma de una función el tipo de dato seguido de los corchetes [] (si es una matriz, se escriben dos corchetes, etc).

No se escribe en la firma el tamaño del array.

Esto no es un inconveniente porque se puede obtener fácilmente el tamaño.

Ejercicio

Implementar el TDA Matriz, que representa una matriz de enteros. Esta clase será en esencia una clase de envoltura (wrapper class) y debe permitir:

1. Instanciar una matriz de tamaño nxm
2. Instanciar una matriz cuadrada de tamaño nxn
3. Modificar y acceder a sus elementos
4. Mostrar por pantalla toda la matriz
5. Otros métodos útiles que consideren (traza, determinante...)

Challenge: es posible implementar una matriz utilizando un array unidimensional.

Esto permite realizar un único acceso a memoria, y dependiendo del lenguaje, puede llegar a ser más eficaz temporalmente. Sin embargo, necesita un manejo más cuidadoso de los índices...

Memoria dinámica

Contenido:
- Estructura de la memoria
- Memoria dinámica
- Referencias
- Garbage collector

Estructura de la memoria

Stack (Pila):
- Variables locales
- Manejo automático

Heap (montículo):
- Objetos y memoria dinámica
- Manejo manual (en C|C++), automático (en Java)

Code segment y Data segment
- Código ejecutable del programa
- Variables globales o estáticas

Memoria dinámica Reserva memoria en tiempo de ejecución

- En C: malloc, free
- En Java: new
- Ventajas: flexibilidad
- Riesgos: fugas de memoria (en lenguajes sin Garbage Collector)

```
String texto = new String( original: "Hola"); // en heap
```

Referencias

Una variable de tipo objeto no guarda el objeto, guarda una referencia (dirección de memoria)
Copiar referencias no es copiar objetos.

```java
int[] a = {1, 2, 3};
int[] b = a;
b[0] = 99;
System.out.println(a[0]);   // imprime 99
```

Recolector de basura (Garbage Collector)

- se encarga de liberar la memoria que ya no se usa.
- Detecta objetos sin referencias
- Ventajas: evita fugas de memoria
- limitaciones: no se controla el momento exacto de liberación.

Práctica - TDA

Escribir un programa que permita manejar una agenda de contactos. Los contactos tienen número de teléfono, nombre y apellido.
El número debe tener 10 dígitos enteros y empezar con "11".
Cada contacto tiene un número único.

Hay limitaciones técnicas para la agenda: no se pueden utilizar estructuras de datos ya implementadas en Java.util para su implementación. El usuario podrá elegir la capacidad máxima de la agenda. Los contactos tienen que estar guardados de forma contigua.

La agenda debe permitir:

1. Agregar un contacto. No se puede agregar un contacto repetido (es decir, con el mismo número) y tampoco se puede agregar si se llena la agenda.
2. Eliminar un contacto por número.
   No se puede eliminar un contacto no existente y tampoco se puede eliminar si la agenda esta vacía.
3. Mostrar todos los contactos en la agenda
4. Cargar y guardar los contactos en un archivo contactos.csv
   El formato del archivo es NÚMERO, NOMBRE, APELLIDO.