Manejo de memoria

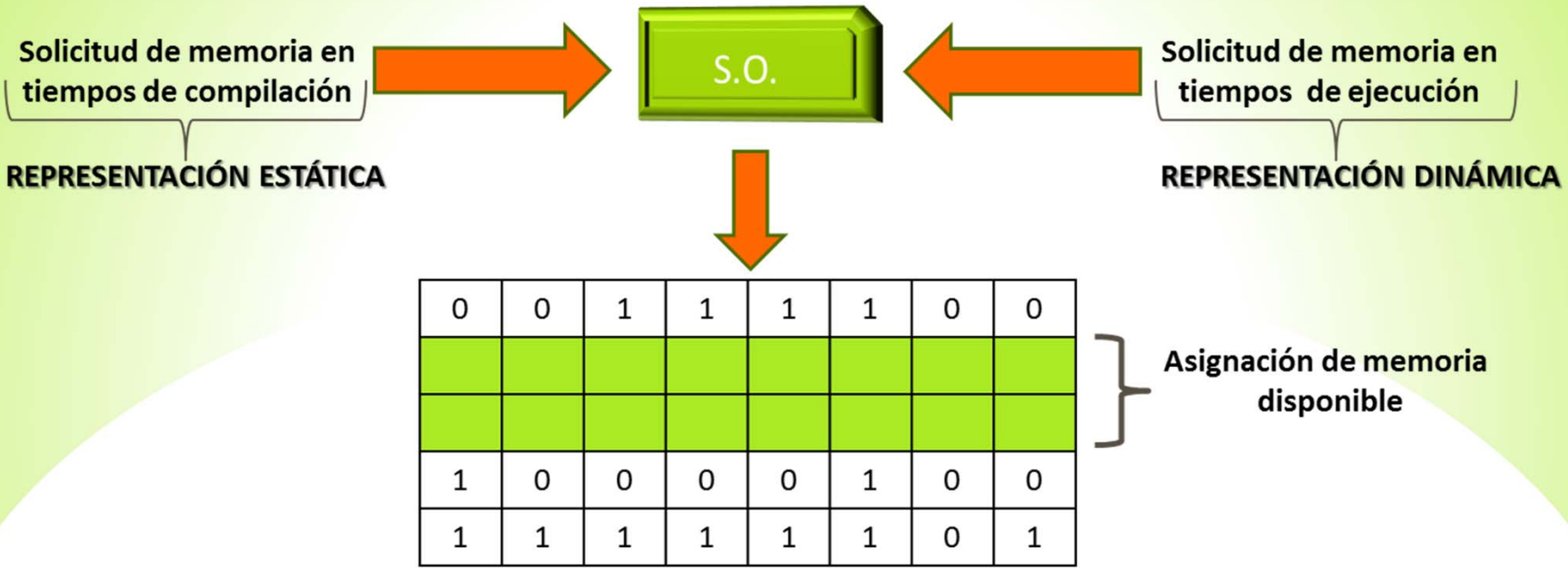
1. Manejo de memoria estática
2. Manejo de memoria dinámica

Introducción

* La administración de memoria de una computadora es una tarea fundamental debido a que la cantidad de memoria es limitada.

* El sistema operativo es el encargado de administrar la memoria del sistema y compartirla entre distintos usuarios y/o aplicaciones.

* El RTS (Run Time System) de un lenguaje de programación administra la memoria para cada programa en ejecución.



* La ejecución de un programa requiere que diversos elementos se almacenen en la memoria:
  * Código del programa (instrucciones)
  * Datos
    * Permanentes
    * Temporales
  * Direcciones para controlar de flujo

el ejecución del programa

Asignación de Memoria Estática y Dinámica

* A la asignación de memoria para algunos *elementos fijos* del programa que es *controlada por el compilador* se le llama **asignación de memoria estática.**

* A la asignación y posible recuperación de memoria durante la *ejecución* de un programa y bajo su control, se le llama **asignación de memoria dinámica.**

Memoria Estática

* Define la cantidad de memoria necesaria para un programa durante el tiempo de compilación.

* El tamaño **no puede** cambiar durante el tiempo de ejecución del programa.

* Algunos lenguajes de programación utilizan la palabra **static** para especificar elementos del programa que deben almacenarse en memoria estática.

Memoria Estática

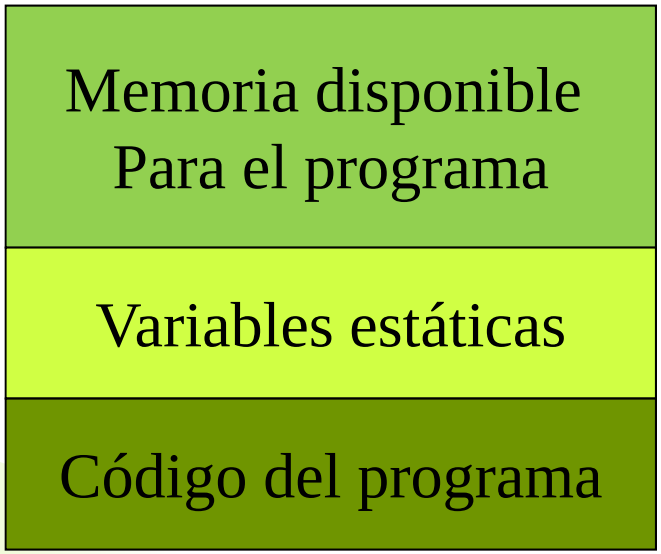
* Elementos que residen en memoria estática:
  * Código del programa
  * Las variables definidas en la sección principal del programa, las cuales pueden solo cambiar su contenido no su tamaño.
  * Todas aquellas variables declaradas como estáticas en otras clases o módulos.

* Estos elementos se almacenan en direcciones fijas que son relocalizadas dependiendo de la dirección en donde el cargador las coloque para su ejecución.

Método común de asignación de memoria

Un **mapa de memoria** (del inglés *memory map*) es una estructura de datos (tablas) que indica cómo está distribuida la memoria. Contiene información sobre el tamaño total de memoria y las relaciones que existen entre direcciones lógicas y físicas, además de poder proveer otros detalles específicos sobre la arquitectura de la computadora.

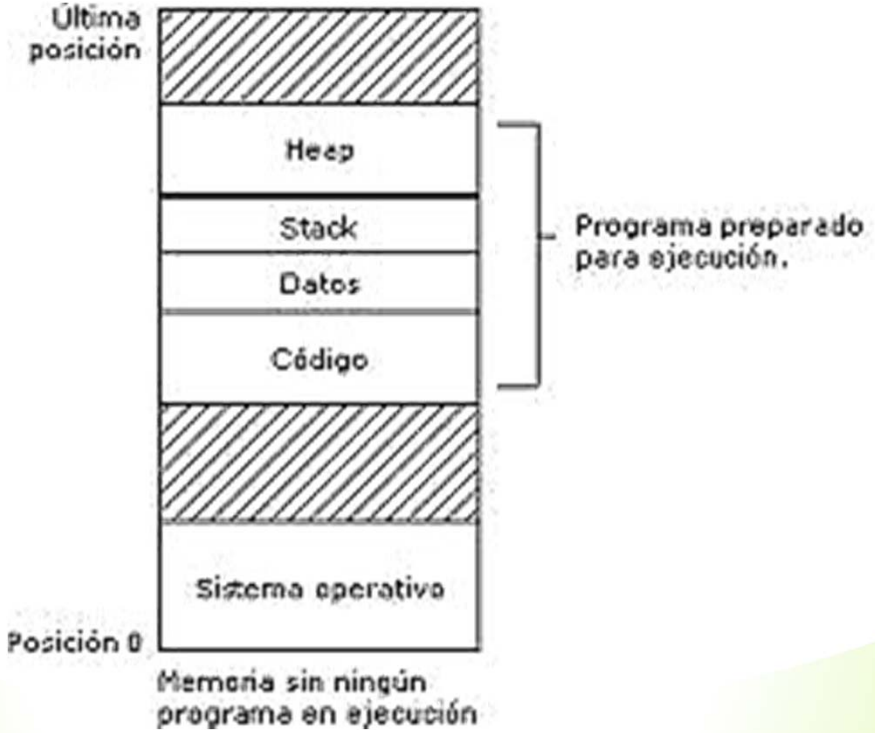
Mapa de memoria



El *stack* de ejecución

* Cada subprograma (procedimiento, función, método, etc.) requiere una representación de si en tiempo de ejecución.

* Estas representaciones se almacenan en el stack de ejecución con el fin de controlar el flujo de ejecución del programa.



Ejemplo

```
....
 public static int factorial (int n){
    if (n==0) return 1;
    else return n*factorial(n-1);
}
public static void main (String[] a){
    int a=5;
    System.out.println(factorial(a));
}
```

¿Cuales elementos del programa serán colocados en memoria estática?

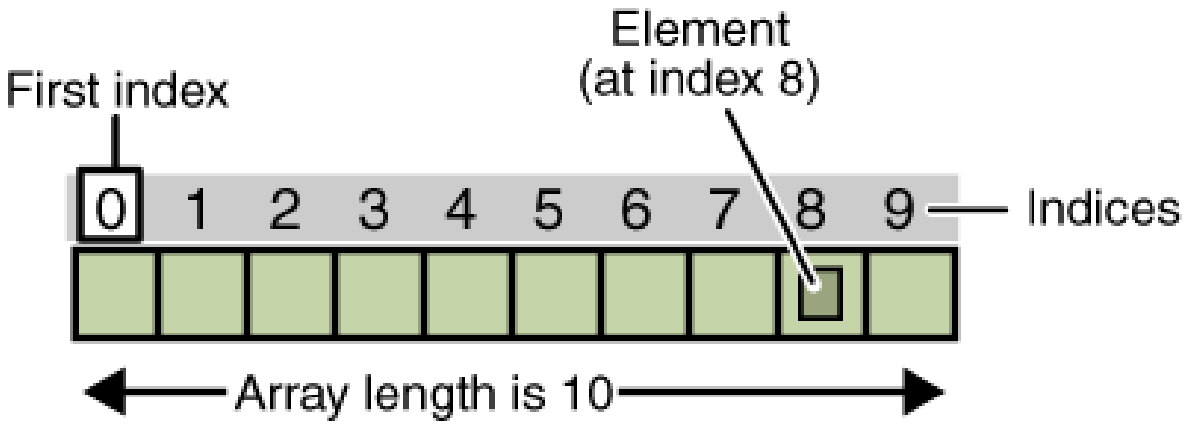
¿Qué elementos se almacenarán en el stack en tiempo de ejecución?

Arreglos

Un arreglo es una estructura de datos que contiene una colección de datos del mismo tipo.

Utilización:

- Temperaturas mínimas de los últimos treinta días
- Valor de las acciones de una empresa durante la última semana



Propiedades de los Arreglos

* Los arreglos se utilizan como contenedores para almacenar datos relacionados
* Todos los datos incluidos en el arreglo son del mismo tipo. Se pueden crear arreglos de enteros (int), flotantes (float), pero en un mismo arreglo no se pueden mezclar datos de tipo int y float
* El tamaño del arreglo se establece cuando se crea el arreglo
* A los elementos del arreglo se accederá a través de la posición que ocupan dentro del conjunto de elementos del arreglo.

Terminología

* Los arreglos unidimensionales se conocen con el nombre de vectores
* Los arreglos bidimensionales se conocen con el nombre de matrices

nombres	Juan	Ana	Marcos	Pablo	Laura
edades	12	21	27	14	21

sueldos

540	540	760
200	220	250
760	760	760
605	799	810

Definición

* Para declarar un arreglo, se utilizan corchetes para indicar que se trata de un arreglo y no de una simple variable del tipo especificado.
* Ejemplo:

Vectores

```
Tipo identificador[];
Tipo[] identificador;
```

Matrices

```
Tipo identificador[][];
Tipo[][] identificador;
```

Creación

* Los arreglos se crean con el operador new

Vector = new tipo [elementos];

* Ejemplo:

float [] notas = new float[ALUMNOS];

int[] temperaturas = new int [7];

Creación

* Los arreglos se crean con el operador new

matriz = new tipo [filas][columnas];

* Ejemplo:

int[][] temperaturas = new int [12][31];

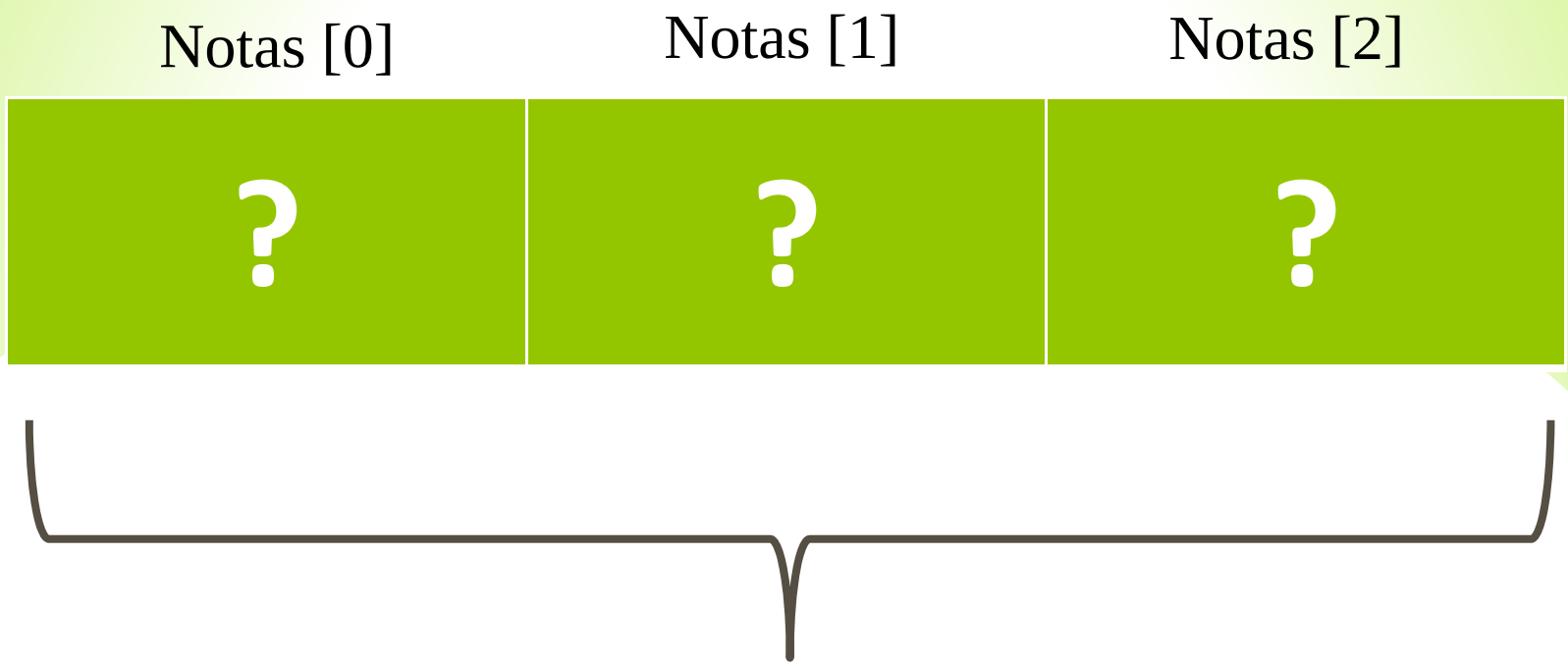
Utilización

* Para acceder a los elementos de un arreglo se utilizan los índices, para indicar la posición del elemento dentro del arreglo
* En Java, el índice de la primera componente de un vector es siempre 0

Vector[índice]

* El tamaño del arreglo puede obtenerse utilizando la propiedad vector.lenght
* Por tanto, el índice del último elemento es vector.length-1
* Ejemplo:

```
float[] notas = new float [3];
```


Notas [0]
Notas [1]
Notas [2]
?
?
?
Notas

Utilización

* Una matriz, es un vector de vectores:
* En Java, el índice de la primera componente de un vector es siempre 0, por lo que matriz[0][0] será el primer elemento de la matriz

matriz[índice1] [índice2];

* El tamaño del arreglo puede obtenerse utilizando la propiedad matriz.lenght
  * Matriz.length nos da el número de filas
  * Matriz[0].length nos da el número de columnas
* Por tanto, el índice del último elemento de la matriz es

matriz[matriz.length-1] [matriz[0].length-1];

Columnas

Filas

(0,0)	(0,1)	(0,2)
(1,0)	(1,1)	(1,2)
(2,0)	(2,1)	(2,2)

Matriz[3][3]

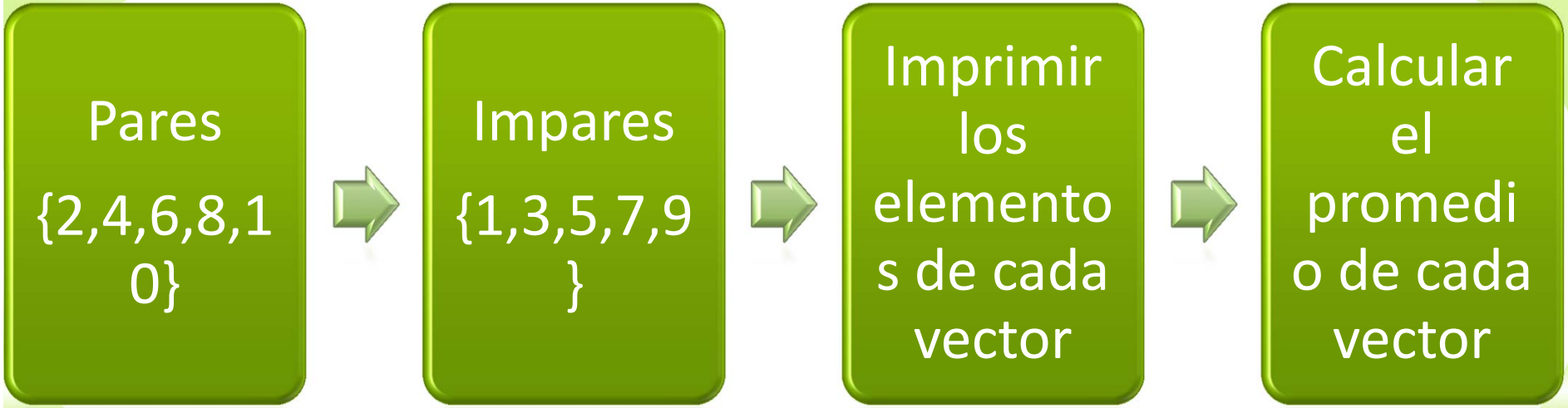
Inicialización en la declaración

* Se puede asignar un valor inicial a los elementos de un arreglo en la propia declaración.

```
int vector[] = {1,2,3,5,7};
int matriz[][] = {{1,2,3}, {4,5,6}};
```

Ejercicio

Crear un programa que muestre 2 vectores y calcule su promedio.



Crear una clase pública llamada Vectores

Dentro del método main() declarar los dos vectores de tipo entero

int pares[] = {2,4,6,8,10};

int impares[]= {1,3,5,7,9};

Dentro del método main crear el método

static void mostrarVector (int datos[])

Dentro del método declarar la variable **i** de tipo entero

Crear un ciclo for con los siguientes parámetros:

for(i=0; i<datos.length; i++)

Dentro del for leeremos cada uno de los elementos que se encuentran dentro del vector utilizando la sentencia datos[i]

Resultado:

```
static void mostrarVector(int datos[])
{ int i;
  for (i=0; i<datos.length; i++)
  System.out.println(datos[i]);
}
```

Crearemos otro método que se llamará media

static float media (int datos[])

Declararemos 3 variables

int i;

int n=datos.length;

int suma=0;

Agregaremos un for como anteriormente lo hicimos y calcularemos la suma de los elementos

suma=suma+datos[i];

Finalmente retornaremos el valor de suma entre el número de elementos

Resultado:

```
static float media (int datos[])
{ int i;
  int n=datos.length;
   int suma=0;
   for (i=0; i<n; i++)
       suma=suma+datos[i];
    return suma/n;
}
```

Una vez creados los dos métodos los mandaremos a llamar en el método principal

Llamaremos a mostrarVector enviándole como parámetros al vector pares

mostrarVector(pares);

Para después mandar el mensaje de la ejecución del método media imprimiendo el promedio calculado

```
System.out.println("Media= " + media(pares));
```

Realizaremos el mismo procedimiento para mandar a llamar al vector impares.

Ejecutemos el programa

☺

```
public class Vectores {

public static void main(String[] args) {

int pares[] = {2,4,6,8,10};
int impares[]= {1,3,5,7,9};

mostrarVector(pares);
System.out.println("Media= " + media(pares));
mostrarVector(impares);
System.out.println("Media= " + media(impares));
}

static void mostrarVector(int datos[])
{  int i;
   for (i=0; i<datos.length; i++)
        System.out.println(datos[i]);
}

static float media (int datos[])
{ int i;
  int n=datos.length;
  int suma=0;

  for (i=0; i<n; i++)
     suma=suma+datos[i];
  return suma/n;
  }
}
```

```
<terminated> Vectores [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe
2
4
6
8
10
Media= 6.0
1
3
5
7
9
Media= 5.0
```

Tarea

* Mostrar 2 ejemplos de utilización de matrices
  1. Mostrar código fuente e imagen de ejecución
  2. Los ejemplos deberán solicitar los datos por parte del usuario
  3. Deberán utilizar métodos