

# Unidad 6

Estructuras de datos estáticas

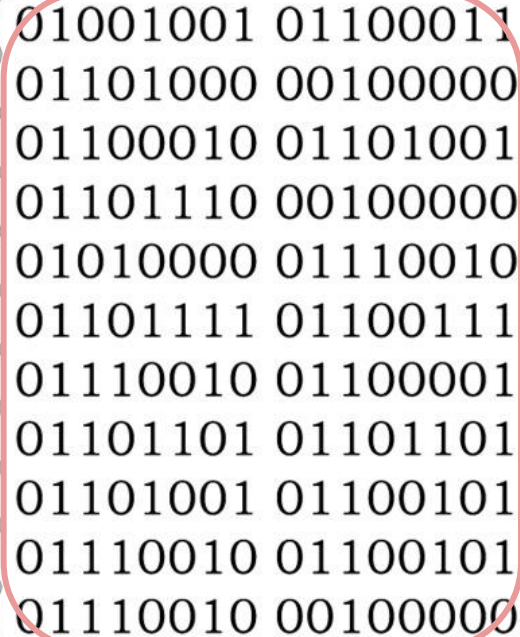
# Tipos simples

```
01001001 01100011  
01101000 00100000  
01100010 01101001  
01101110 00100000  
01010000 01110010  
01101111 01100111  
01110010 01100001  
01101101 01101101  
01101001 01100101  
01110010 01100101  
01110010 00100000
```

(int) 1.746.952.809

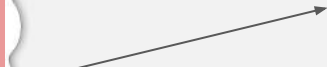
(char) "m"

# Tipos complejos



01001001	01100011
01101000	00100000
01100010	01101001
01101110	00100000
01010000	01110010
01101111	01100111
01110010	01100001
01101101	01101101
01101001	01100101
01110010	01100101
01110010	00100000

(Persona) Nombre,  
apellidos, edad...



# Estructuras de datos

Los medios para manejar grandes cantidades de información de manera eficiente:

- Estructuras de datos **estáticas**: son aquellas en las que el tamaño ocupado en la memoria se define antes de que el programa se ejecute y no puede ser modificado durante la ejecución del programa.
- Estructuras de datos **dinámicas**: son aquellas en las que el tamaño de memoria se define en tiempo de ejecución en función de las necesidades del programa.

# Estructuras de datos estáticas

Array, arreglo, vector

Tipo enumerado

# Array

Permite almacenar una ristra de datos de un mismo tipo. El tamaño de los arrays se declara en un primer momento y no puede cambiar en tiempo de ejecución.

```
tipo_dato nombre_array[];  
nombre_array = new tipo_dato[tamano];
```

Por ejemplo, para crear un array de 10 elementos de tipo char

```
char arrayCaracteres[];  
arrayCaracteres = new char[10];
```

# Inicialización de arrays

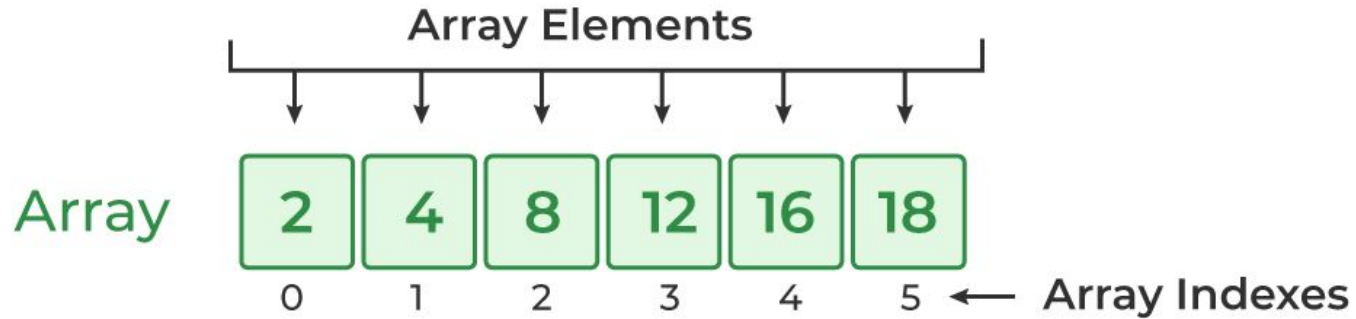
```
float floats[] = new float[5];
```

```
char[] chars = new char[5];
```

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
```

```
int[] myNum = {10, 20, 30, 40};
```

# Estructura de un array



```
int[] myNum = {2, 4, 8, 12, 16, 18};
```

```
System.out.println(myNum[3]);
```

```
myNum[5] = 13;
```



# Recorrer un array

```
String[] cars = { "Volvo", "BMW", "Ford", "Mazda" };  
for (int i = 0; i < cars.length; i++) {  
    System.out.println(cars[i]);  
}
```

Recorreremos siempre hasta length-1

# Array con tipos complejos

```
Persona persona1 = new Persona();
```

```
Persona persona2 = new Persona();
```

```
Persona persona3 = new Persona();
```

```
Persona[] personas = {persona1, persona2, persona3};
```

# Array con tipos complejos

```
Persona[] personas = new Persona[150];  
for (int i = 0; i < 150; i++) {  
    Persona p = new Persona();  
    p.setNombre("persona"+i);  
    personas[i]=p;  
}
```

# Foreach

```
Persona[] personas = new Persona[150];  
for (Persona persona : personas) {  
    persona=new Persona();  
    persona.setNombre("Persona");  
}
```

# Arrays multidimensionales (matrices)

```
int[][] matrizVacía = new int[2][3];  
int[][] matriz = {{115,12,3},{65,23,9}};  
System.out.println(matriz[0][2]);
```

	0	1	2
0	115	12	3
1	65	23	9

# Tipo enumerado

```
public class Main {  
    enum Level {  
        LOW,  
        MEDIUM,  
        HIGH  
    }  
  
    public static void main(String[] args) {  
        Level myVar = Level.MEDIUM;  
        System.out.println(myVar) ;  
    }  
}
```