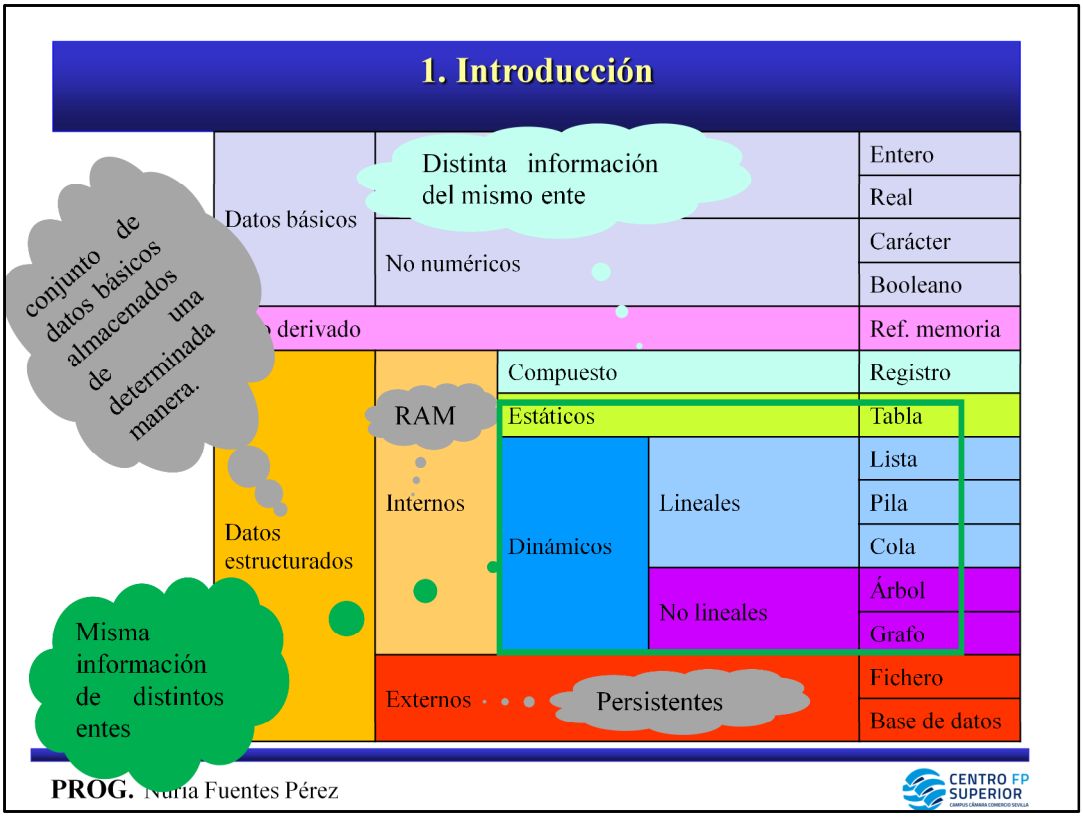
UD10. ESTRUCTURAS ESTÁTICAS DE DATOS.

1. Introducción.
2. Registros.
3. Tablas o arrays.
   A. Tablas unidimensionales o vectores.
      a. Operaciones sobre un vector.
   B. Algoritmos de ordenación.
   C. Tabla bidimensional o matriz.
   D. Tablas compuestas vs paralelas.
      a. Operaciones sobre tabla compuesta.

PROG. Nuria Fuentes Pérez


1. Introducción
Distinta información del mismo ente
Datos básicos
No numéricos
Entero
Real
Carácter
Booleano
derivado
Ref. memoria
conjunto de datos básicos almacenados de una determinada manera.
Compuesto
Registro
RAM
Estáticos
Tabla
Internos
Dinámicos
Lineales
Lista
Pila
Cola
Datos estructurados
No lineales
Árbol
Grafo
Misma información de distintos entes
Externos
Persistentes
Fichero
Base de datos
PROG. Nuria Fuentes Pérez
CENTRO FP SUPERIOR

1. Introducción

El uso de estructuras facilita al programador el manejo de los datos.

Se pueden considerar de forma general, los siguientes datos:

- Datos **internos**: son los que residen en la memoria principal del ordenador.
- Datos **externos**: son los que residen en una memoria auxiliar, externos a la memoria principal.
- Datos **estáticos**: son aquellos cuyo tamaño queda definido en la creación de la estructura y no se puede modificar durante el uso de la misma.
- Datos **dinámicos**: son aquellos cuyo tamaño puede ser modificado durante el uso de la estructura.

1. Introducción

- Datos **compuestos**: los registros o estructuras son datos internos formados por un conjunto de tipos básicos y derivados. (distintas informaciones del mismo ente) A diferencia de los no compuestos que contienen misma información de distintos entes.

- Datos **lineales**: son los que pueden estar enlazados sólo con un elemento anterior y con un elemento posterior.

- Datos **no lineales**: son los que pueden enlazarse con más de un elemento anterior y más de un elemento posterior.

2. Registros.

Es una estructura de datos (conjunto de datos básicos) que se almacenan en posiciones consecutivas de memoria y que la relación que tienen entre sí es ser **distintas informaciones del mismo ente**.

NOMBRE		
NOM	APE1	APE2
XXXXXXXXXX	XXXXXXXXXXXXXX	XXXXXXXXXXXXXX

2. Registros.

Pseudocódigo

```
nRegistro es Registro compuesto de
   nCampo1 <tipoDeDatos>
   nCampo2 <tipoDeDatos>
   ...
Fin Registro
```

```
curso es Registro compuesto de
   codCurso es numérico entero
   nota es numérico real
Fin Registro
```

Campos: se puede usar igual que cualquier variable de tipo básico.

Registro: sólo se puede usar en asignaciones.
En algunos lenguajes también en comparaciones

2. Registros.

En el paradigma de POO, los registros serían……

Objetos

Los registros permiten una abstracción de los datos básicos que los componen mientras que los objetos permiten no solo una abstracción de los atributos que contienen sino también de los algoritmos que permiten su uso.

3. Tablas o arrays.

o**Tabla**: es una estructura de datos constituida por un número fijo de elementos, todos ellos del mismo tipo, almacenados en posiciones consecutivas de memoria y que tienen una relación entre sí.

o**Elemento**: cada uno de los datos que forman parte integrante de una tabla.

o**Nombre de la tabla**: el identificador utilizado para referenciar la tabla y, de forma global, los elementos que la forman.

o**Tipo de tabla**: marca el tipo de dato básico que es común a todos y cada uno de los elementos que forman la tabla. (entero, real, lógico, etc.)

o**Subíndice o índice**: es una valor numérico entero y positivo a través del cual podemos acceder directamente a los distintos elementos de la tabla, pues marca la situación relativa de cada elemento dentro de la misma.

3. Tablas o arrays.

o**Tamaño de la tabla**: el tamaño o longitud de una tabla viene determinado por el número máximo de elementos que la forman, siendo el tamaño mínimo 1 elemento.

o**Acceso a un elemento de la tabla**: cada elemento de la tabla se trata de un dato simple de cualquiera de los tipos ya conocidos. La forma de acceder a cada uno de esos elementos es mediante el nombre de la tabla seguido del índice correspondiente entre paréntesis. (Inicio en 0 o 1).

o**Dimensión de la tabla**: viene determinado por el número de índices que necesitamos para acceder a cualquiera de los elementos que forman la tabla.

3.A. Tablas unidimensionales o vectores.

SUELDO			
1	2	...	15
9999.99	9999.99	...	9999.99

sueldo(4)

- Tabla
- Elemento
- Nombre de la tabla
- tipo de la tabla
- Subíndice
- tamaño

3.A.a. Operaciones sobre un vector.

✓Definición

nTabla es Tabla (tamaño) tipo
Acceso a un elemento:
nTabla(i) ➔ 0..tamaño-1

tamaño: (nº elementos, último índice)

Índices: 0..nºelementos-1
1..nºelementos

Ejemplo:
num es Tabla(5) es numérico entero

✓Carga

Hacer que los elementos de la tabla tengan un valor, la inicialización es una manera de cargar la tabla. Se puede cargar también desde teclado o de cualquier otra forma.

Se usará un **Para** si vamos a cargar todos los elementos o cualquiera de las otras repetitivas si puede que no se cargue completamente, en este caso sería conveniente inicializarla primero.

```
num(0) ← 5
num(1) ← 23
num(2) ← 8
num(3) ← 12
num(4) ← 7
```

```
Para i de 0 a 4 hacer
   Escribir "Introduzca un nº: "
   Leer num(i)
Fin Para
```

3.A.a. Operaciones sobre un vector.

- ✓**Definición**: declarar identificador para el vector.
- ✓**Acceso**: usar el valor de un elemento del vector.
- ✓**Carga**: dar un valor a los elementos del vector.
- ✓**Recorrido**: moverse por los distintos valores del vector.
- ✓**Búsqueda**: buscar el elemento del vector con el valor deseado.
  - Vector desordenado
    - Secuencial
  - Vector ordenado
    - Secuencial
    - Dicotómica
- ✓**Ordenación**: recolocar los valores en el vector según un criterio.

3.A.a. Operaciones sobre un vector. JAVA

✓**Definición**

```
//Entorno:
   tipo[] nTabla; //esta mejor
   tipo nTabla[];
//Algoritmo:
   nTabla=new tipo[tamaño];
```

```
//Entorno:
   tipo[] nTabla={valor1,valor2,…,valorN};
```

Ejemplo:
num es Tabla(5) es numérico entero

Ejemplo:
int[] num;
num=new int[5];

✓**Acceso**

nTabla[i] ➔ i →0..tamaño-1

3.A.a. Operaciones sobre un vector. JAVA

✓Carga

Hacer que los elementos de la tabla tengan un valor, la inicialización es una manera de cargar la tabla. Se puede cargar también desde teclado o de cualquier otra forma.

Se usará un **Para** si vamos a cargar todos los elementos o cualquiera de las otras repetitivas si puede que no se cargue completamente, en este caso sería conveniente inicializarla primero.

```
Entorno:
   num es tabla(5) es numérico entero
Algoritmo:
   num(0) ← 5
   num(1) ← 23
   num(2) ← 8
   num(3) ← 12
   num(4) ← 7
```

```
//Entorno:
   int[] num;
//Algoritmo:
   num=new int[5];
   num[0] = 5;
   num[1] = 23;
   num[2] = 8;
   num[3] = 12;
   num[4] = 7;
```

≈

```
//Entorno:
   int[] num={5,23,8,12,7};
```

```
Para i de 0 a 4 hacer
   Escribir "Introduzca un nº: "
   Leer num(i)
Fin Para
```

```
for(i=0;i<=4;i++){
   System.out.print("Introduzca un nº: ");
   num[i]=Leer.datoInt();
}//Fin Para
```

3.A.a. Operaciones sobre un vector. JAVA

✓Recorrido

Supone tratar todos y cada uno de los elementos de una tabla bien de izquierda a derecha o de derecha a izquierda. En cualquier caso se usará un Para.

```
Para i de 0 a 4 hacer
   Escribir num(i)
Fin Para
```

```
for(i=0;i<=4;i++){
   System.out.println(num[i]);
}
```

```
for(i=0;i<num.length;i++){
   System.out.println(num[i]);
}
```

```
//Entorno:
   int[] m1 = {10, 20, 30, 40, 50};
   int[] m2;
//Algoritmo:
   m2 = (int[])m1.clone();
   if (m1.equals(m2)){
         System.out.println("m1 y m2 se refieren a la misma matriz");
   }else{
         System.out.println("m1 y m2 se refieren a matrices diferentes");
   }//Fin Si
```

3.A.a. Operaciones sobre un vector.

✓**Búsqueda secuencial vector desordenado**

55	0	25	4	16	3	85	2
0	1	2	3	4	5	6	7

- Buscar el primero que aparezca.
- Buscar el último que aparezca.
- Buscar todas las apariciones.

✓**Búsqueda secuencial vector ordenado**

0	2	3	4	16	25	55	85
0	1	2	3	4	5	6	7

ASCENDENTE

85	55	25	16	4	3	2	0
0	1	2	3	4	5	6	7

DESCENDENTE

3.A.a. Operaciones sobre un vector.

✓**Búsqueda dicotómica**

•Dividimos la tabla por la mitad y comprobamos si el elemento central es o no el buscado.

•En caso de no serlo, comprobamos si el elemento buscado es mayor o menor que el elemento central.

•Según sea mayor o menor, cogeremos la mitad izquierda o derecha de la tabla respectivamente.

•Este segmento de tabla la tratamos como una tabla independiente y volvemos a dividirlo por la mitad igual que antes, continuando con el mismo proceso.

•Esto lo repetiremos hasta que sólo nos quede un elemento o encontremos el elemento buscado.

1	2	3	6	10	13	20	27	28
0	1	2	3	4	5	6	7	8

3.B. Operaciones sobre un vector.

➢Ordenación. Método de intercambio

Consiste en ir colocando en pasos sucesivos el elemento menor de la lista en las primeras posiciones. Así, en el primer paso, el elemento más pequeño ocupará la primera posición, en el segundo paso el segundo elemento más pequeño ocupará la segunda, etc.

5	1	1	1	1	1	1
10	10	5	5	5	5	5
8	8	**8**	8	8	8	8
22	22	22	22	10	10	10
1	**5**	10	**10**	22	12	12
12	12	12	12	**12**	22	15
15	15	15	15	15	**15**	22

Tabla de "*N*" ➔ *N-1* paso para ordenarla.

El hecho de no ejecutarse ningún cambio en un paso no indica que la tabla esté ordenada.

Hemos hecho ordenación ascendente buscando el pequeño. También podría hacerse buscando mayor y ponerlos en las últimas posiciones. O bien, de forma descendente por los dos métodos (el menor en últimas posiciones o el mayor en primeras posiciones).

3.B. Operaciones sobre un vector.

Ordenación. Método de burbuja

Este método consiste en ir desplazando el elemento más grande de la tabla a las últimas posiciones de ésta mediante sucesivas comparaciones de las parejas: 1-2, 2-3, 3-4, 4-5, ...

PASO 1						
22	13	13	13	13	13	13
13	**22**	17	17	17	17	17
17	17	**22**	12	12	12	12
12	12	12	**22**	7	7	7
7	7	7	7	**22**	15	15
15	15	15	15	15	**22**	2
2	2	2	2	2	2	**22**

	PASO 1	PASO 2	PASO 3	PASO 4	PASO 5	PASO 6
22	13	13	12	7	7	2
13	17	12	7	12	2	7
17	12	7	13	2	12	12
12	7	15	2	13	13	13
7	15	2	15	15	15	15
15	2	17	17	17	17	17
2	22	22	22	22	22	22

Tabla de *N* elementos ➔ *N-1* paso para ordenarla.

Si en un paso no se efectúan cambios, se puede decir que la tabla está clasificada.

Hemos hecho ordenación ascendente poniendo el mayor en las últimas posiciones, también puede hacerse poniendo el menor en las primeras posiciones, o bien, ordenado de forma descendente por los dos métodos (menores al principio o mayores al final).

3.C. Tabla bidimensional o matriz.

		Alumnos									
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Matemáticas	**0**	9	4	8.25	6	10	2.25	1.75	7.25	3	5.5
Física	**1**	7.75	7	9.25	2.5	5	6	3.5	6.75	1	4
Historia	**2**	5.25	.75	8.5	7	6.5	10	0.25	8.25	8	7.5
Filosofía	**3**	5.5	0.25	7	8.5	8	4.75	9	8	6.5	6

- **Nombre de la tabla**: nota.
- **Tipo**: numérico real.
- **Tamaño**: 40 elementos.
- **Elementos de la tabla**: nota(0,0), nota(0,1), nota(0,2), ..., nota(0,9), nota(1,0), nota(1,1), ..., nota(1,9), ..., nota(3,0), ..., nota(3,9)
- **Índices**: fila:0, 1, 2, 3 y columna:0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- **Dimensión**: 2

3.C.a. Operaciones sobre una matriz.

✓Definición

```
nTabla es Tabla (filas,columnas) tipo
Acceso a un elemento:
   nTabla(f,c) ➔ 0..tamaño-1
```

```
filas:  (nº filas,  último índice fila)
Columnas: (nº columnas, último índice columnas)
Índices:      0..nºelementos-1
              1..nºelementos
```

✓Carga

Ejemplo:
m es Tabla(2,3) es numérico entero

Hacer que los elementos de la tabla tengan un valor, la inicialización es una manera de cargar la tabla. Se puede cargar también desde teclado o de cualquier otra forma.

Se usará un **Para** si vamos a cargar todos los elementos o cualquiera de las otras repetitivas si puede que no se cargue completamente, en este caso sería conveniente inicializarla primero.

```
m(0,0) ← 1
m(0,1) ← 2
m(0,2) ← 3
m(1,0) ← 4
m(1,1) ← 5
m(1,2) ← 6
```

```
Para fil de 0 a 1 hacer
   Para col de 0 a 2 hacer
      Escribir "Introduzca un nº ("; fil; ","; col; "): "
      Leer m(fil,col)
   Fin Para
Fin Para
```

3.C.a. Operaciones sobre una matriz. JAVA

✓Definición

```
//Entorno:
    tipo[][] nMatriz; //esta mejor
    tipo nMatriz[][];
//Algoritmo:
    nMatriz=new tipo[filas][columnas];
```

```
//Entorno:
    tipo[][] nMatriz={{valor00,…},{valor10,…}…};
```

Ejemplo:
m es Tabla(2,3) es numérico entero

```
Ejemplo:
int[][] m;

m=new int[2][3];
```

✓Acceso

nTabla[fila][columna] ➔ fila →0..filas-1
columnas →0..columnas-1

3.C.a. Operaciones sobre una matriz. JAVA

✓Carga

Hacer que los elementos de la tabla tengan un valor, la inicialización es una manera de cargar la tabla. Se puede cargar también desde teclado o de cualquier otra forma. Si puede que no se cargue completamente, en este caso sería conveniente inicializarla primero.

```
Entorno:
   m es tabla(2,3) es numérico entero
Algoritmo:
   m(0,0) ← 1
   m(0,1) ← 2
   m(0,2) ← 3
   m(1,0) ← 4
   m(1,1) ← 5
   m(1,2) ← 6
```

```
//Entorno:
   int[][] m;
//Algoritmo:
   m=new int[2][3];
   m[0][0] = 1;
   m[0][0] = 1;
   m[0][1] = 2;
   m[0][2] = 3;
   m[1][0] = 4;
   m[1][1] = 5;
   m[1][2] = 6;
```

≈

```
//Entorno:
   int[][] m={{1,2,3},{4,5,6}};
```

```
Para fil de 0 a 1 hacer
   Para col de 0 a 2 hacer
      Escribir "Introduzca un nº (", fil, ", ", col, "): "
      Leer m(fil,col)
   Fin Para
Fin Para
```

```
for(fil=0;fil<m.length;fil++){
   for(col=0;col<m[0].length;col++){
      System.out.print("Introduzca un nº ("+ fil+", "+col+"): ");
      m[fil][col]=Leer.datoInt();
   }//Fin Para
}//Fin Para
```

3.D. Tablas compuestas vs. tablas paralelas

ALUMNOS							
ALUMNOS(0)		ALUMNOS(1)		...	ALUMNOS(6)		ALUMNOS(7)	
COD	NOTA	COD	NOTA		COD	NOTA	COD	NOTA
151	9.63	25	2.5		96	6.25	55	1.25

- **Nombre de la tabla**: ALUMNOS
- **Tipo**: compuesta de código y nota del alumno.
- **Tamaño**: 8 elementos.
- **Elementos de la tabla**: alumnos(0), ...alumnos(7)//cod(0),nota(0)
- **Índices**: 0, ...7

COD				
151	25	...	96	55
0	1		6	7

NOTA				
9.63	2.5	...	6.25	1.25
0	1		6	7

3.D. Tablas compuestas vs. tablas paralelas

	COD	NOTA
0	151	9.63
1	25	2.5
2	⋮	⋮
3		
4		
5		
6	96	6.25
7	55	1.25

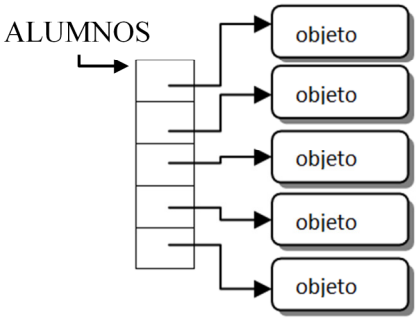
	COD
0	151
1	25
2	⋮
3	
4	
5	
6	96
7	55

	NOTA
0	9.63
1	2.5
2	⋮
3	
4	
5	
6	6.25
7	1.25

NO ES UNA MATRIZ
Ningún dato se repite 16 veces

3.D. Tablas compuestas en POO

ALUMNOS							
ALUMNOS(0)		ALUMNOS(1)		...	ALUMNOS(6)		ALUMNOS(7)	
COD	NOTA	COD	NOTA		COD	NOTA	COD	NOTA
151	9.63	25	2.5		96	6.25	55	1.25



3.D.a Tablas compuestas en Java

✓**Definición**

```
Entorno:
   nTabla es Tabla (tamaño) de nReg
   nRegistro es Registro compuesto de
      nCampo1 <tipoDeDatos>
      nCampo2 <tipoDeDatos>
        ...
   Fin Registro


Algoritmo:
//Acceso a un elemento:
   nTabla(i)
   nCampo1(i)
```

```
//Entorno:
   ObjReg[] nTabla;
//Algoritmo:
   nTabla=new objReg[tamaño];
//OJO– crear objeto
   nTabla[i]=new objReg();

Acceso a un elemento:
   nTabla[i]
   nTabla[i].nMiembro
```

✓**Carga**

✓**Recorrido**

✓**Búsqueda**

✓**Ordenación**

3.D.a Tablas compuestas en Java
Ejemplo

```
//Entorno:
   Yeti[] misYetis;
//Algoritmo:
   misYetis=new Yeti[3];
//OJO– crear objeto
   misYetis[i]=new Yeti();

Acceso a un elemento:
   misYetis[i] → es un objeto Yeti
   misYetis[i].alimentarYeti();
```