Programación I

Tema 5: Estructuras de datos básicas

Contenido

- Tablas: Listas y Matrices
- Registros

Motivación

- No se pueden resolver los siguientes problemas con las herramientas que disponemos hasta ahora

  Ej1. Diseña e implementa un programa, que dada una secuencia de N números, la muestre en orden invertido

  Ej2. Diseña e implementa un programa, que dada una secuencia de N números, muestre el numero que más se repite

  Ej3. Diseña e implementa un programa que muestre el nombre y el DNI del alumno con mejor nota de clase y la nota media de la clase

Motivación

- Los tipos de datos disponibles no nos sirven.
  - Representan datos simples (*primitivos* o *escalares*)
- Necesitamos almacenar varios datos del mismo tipo (Ejemplo1, Ejemplo2)
- Necesitamos almacenar datos de tipos distintos relacionados entre si (Ejemplo3)

Motivación

- **Tipos de datos estructurados:** Agrupan varios valores
  - **Tabla:** Agrupa valores **del mismo tipo**
  - **Registro:** Agrupa valores (de tipos distintos) **relacionados entre sí**

Tabla: Definición

- Para definir una tabla se necesitan los siguientes elementos:
  - **Identificador** para referenciar la tabla
  - **Número de elementos**
  - **Tipo de datos** de los elementos de la tabla
- Ejemplo

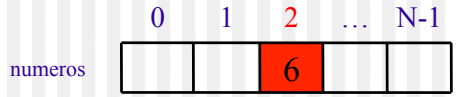
```
int numeros[];          // define una variable para manipular tablas
numeros = new int[5];   // Crea una tabla de 5 elementos
```

- Utilizar el operador **new** para crear la tabla y reservar el espacio de memoria necesario.

Acceso a los elementos de la tabla

- Para acceder a un elemento de una tabla hay que indicar la tabla (**identificador**) y el lugar que ocupa el elemento dentro de la tabla (**índice**)

numeros[2]



Acceso a los elementos de la tabla

- Se puede utilizar una expresión para indicar el valor del índice
  - numeros[i] /* Se accede al I-ésimo elemento de la tabla teniendo en cuenta el valor de I en el momento de ejecución */
  - numeros[2*i-3] /* Se calcula el resultado de la expresión en tiempo de ejecución. SI el valor de I es 4, se accede al **elemento 5 de la tabla** */
- CUIDADO!! El valor del índice no puede estar fuera del rango indicado en la definición

Tabla: Operaciones

- Cada elemento de la tabla contiene un valor del tipo especificado en la definición de la tabla
- Cada elemento funciona como una variable simple (primitiva) del mismo tipo

Tabla: Operaciones

- Acceso al valor de un elemento de una tabla

var1 = numeros[1];

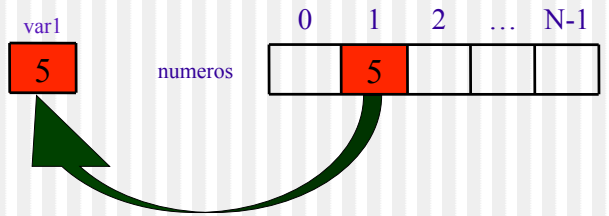


Tabla: Operaciones

- Asignación de un valor a un elemento de la tabla

numeros[1] = var2;

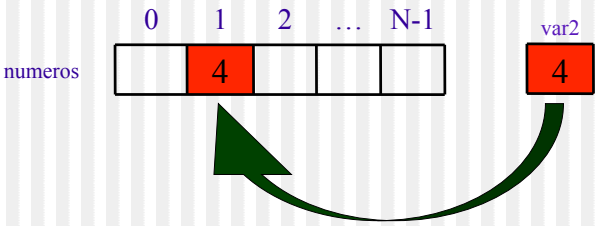
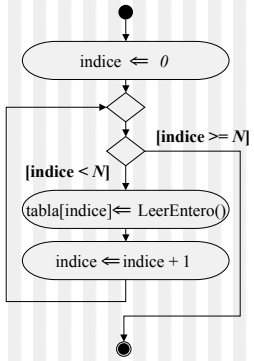


Tabla: operaciones basicas

- **Inicializar** una tabla
- **Visualizar** en pantalla los valores de una tabla
- **Copiar** los valores de una tabla a otra tabla
- **Calcular el** valor **minimo** en una tabla
- **Calcular el** valor **maximo** en una tabla
- Dado un valor **buscar** si dicho valor se encuentra en la tabla
- **Ordenar** los valores de una tabla de menor a mayor
- **Ordenar** los valores de una tabla de mayor a menor

Tabla: operaciones

- **Inicializar** una tabla



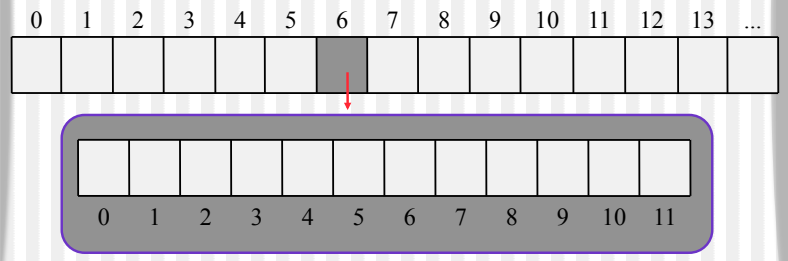

```
private static void inicializarTabla (int[] tabla)
{
  BufferedReader ent = new BufferedReader(new
                                InputStreamReader(System.in));
  try{
    for (int indice=0;indice<tabla.length;indice++)
    {
      tabla[indice] = Integer.parseInt(entrada.readLine();
    }
  }catch(Exception e){….}
}
```


Tablas bidimensionales: Matrices

- A veces nos interesa que los elementos de una tabla, a su vez, sean también tablas
- Ej. Diseña e implementa un programa que calcule las medias de las lluvias de los países de la Comunidad Económica Europea y muestre el mes de más lluvias de cada uno de los países
  - Necesitamos la distribución de lluvias por meses para cada uno de los países.

Tablas bidimensionales: Matrices

- tablaLluviasCEE

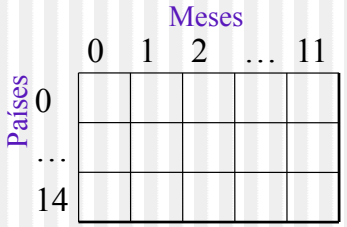


Tablas Bidimensionales: Matrices

- Definición de una matriz

```
float tablaLluviasCEE[][];
tablaLluviasCEE = new float[15][12];
```

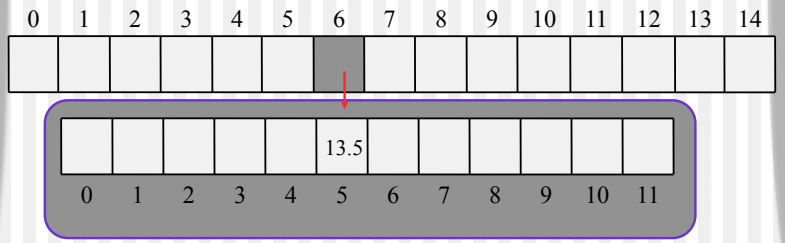
- Se debe indicar el número de elementos para:
  - Filas: nº países (15)
  - Columnas: nº meses (12)



Tablas Bidimensionales: Matrices

- Acceso a un elemento de la matriz
  - Hay que indicar la fila y la columna en la que se encuentra
  - Dos índices

tablaLluviasCEE[**6**][**5**] = 13.5;

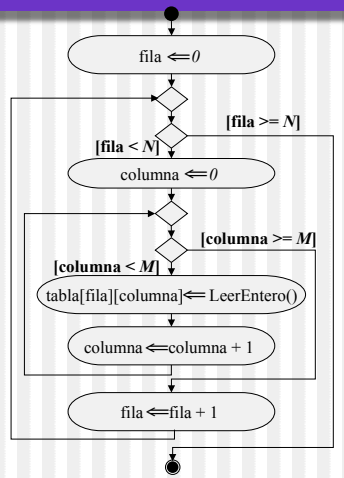


Tablas Bidimensionales: Matrices

- Inicialización
  - Para inicializar una tabla de dos dimensiones hay que asignar un valor para cada uno de los elementos.
  - Dos maneras distintas
    - Por filas
    - Por Columnas

Tablas Bidimensionales: Matrices

- Recorrido por filas de una matriz de N*M



Tablas Bidimensionales: Matrices

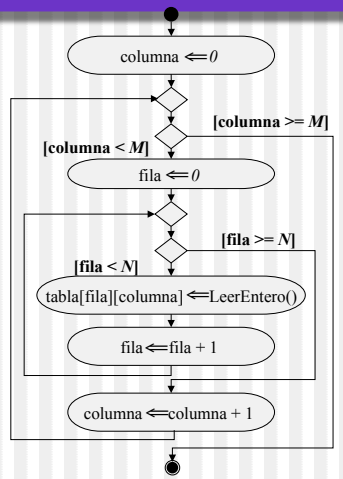
- Recorrido por filas

```
private static void inicializarPorFila(int[][]tabla)
{
   BufferedReader ent = new BufferedReader(new InputStreamReader(System.in));4
   try{
         for (int fila = 0; fila < tabla.length;fila++)
         {
            for (int columna = 0; columna < tabla[fila].length; columna++)
            {
               tabla[fila][columna] = Integer.parseInt(entrada.readLine());
            }
         }
      } catch(Exception e){…}
}
```

Tablas Bidimensionales: Matrices

- Recorrido por Columnas de una matriz de N*M

Cambia el orden de los bucles



Tablas Bidimensionales: Matrices

- Recorrido por Columnas

```
Private static void inicializarPorFila(int[][]tabla)
{
  BufferedReader ent = new BufferedReader(new InputStreamReader(System.in));
  try{
      for (int columna = 0; columna < tabla[0].length;columna++)
      {
         for (int fila = 0; fila < tabla.length; fila++)
         {
            tabla[fila][columna] = Integer.parseInt(entrada.readLine());
         }
      }
    }catch(Exception e){…}
}
```


Registros

- Tipo de datos definido por el usuario
- Datos de distintos tipos relacionados entre si
- Cada dato se almacena en un campo (atributo)

```
public class TipoAlumno
{
  public String dni;
  public String nombre;
  public String apellido;
  public double nota;
}
```

- **Se deben definir en ficheros independientes**

Registros

- Definición de una variable de tipo registro

```
TipoAlumno alum1;
```

- Creación de un nuevo registro: *new*

```
alum1 = new TipoAlumno();
```

- Acceso a los campos del registro

```
alum1.dni = "72.354.623"
System.out.println(alum1.dni);
```

Ejemplo: ImagenPPM

```
public class ImagenPPM {
    public int ancho;
    public int alto;
    public int numColores;
    public Pixel[][] imagen;
}

public class Pixel {
    public int rojo;
    public int verde;
    public int azul;
}
```


Ejemplo: Leer una imagen (I)

```
public static ImagenPPM cargarImagen(String pFichero) throws Exception {
    ImagenPPM imagenCargada = new ImagenPPM();
    BufferedReader entrada = null;
    entrada = new BufferedReader(new FileReader(pFichero));
    // Leer codigo (P3)
    String linea = entrada.readLine();
    linea = entrada.readLine();
    while (linea.startsWith("#")) {
        linea = entrada.readLine();
    }
```


Ejemplo: Leer una imagen (I)

```
public static ImagenPPM cargarImagen(String pFichero) throws Exception {
    ImagenPPM imagenCargada = new ImagenPPM();
    BufferedReader entrada = null;
    entrada = new BufferedReader(new FileReader(pFichero));
    // Leer codigo (P3)
    String linea = entrada.readLine();
    linea = entrada.readLine();
    while (linea.startsWith("#")) {
        linea = entrada.readLine();
    }
```


Ejemplo: Leer una imagen (II)

```
// Obtener las dimensiones
String dims[] = linea.split(" ");
imagenCargada.ancho = Integer.parseInt(dims[0]);
imagenCargada.alto = Integer.parseInt(dims[1]);
// Leer numColores
imagenCargada.numColores = Integer.parseInt(entrada.readLine());
// Crear la matriz de pixeles
imagenCargada.imagen = new Pixel[imagenCargada.alto][imagenCargada.ancho];
```

Ejemplo: Leer una imagen (II)

```
// Obtener los pixeles
for (int i = 0; i < imagenCargada.alto; i++) {
        for (int j = 0; j < imagenCargada.ancho;j++) {
                imagenCargada.imagen[i][j] = new Pixel();
                imagenCargada.imagen[i][j].rojo = Integer.parseInt(entrada.readLine());
                imagenCargada.imagen[i][j].verde = Integer.parseInt(entrada.readLine());
                imagenCargada.imagen[i][j].azul = Integer.parseInt(entrada.readLine());
                }
        }
entrada.close();
return imagenCargada;
}
```


Tablas de tipos definidos por el usuario

- Se pueden definir tablas de registros definidos por el usuario

  TipoAlumno tablaAlumnos[];

  tablaAlumnos = new TipoAlumno[10];

- Creación de un nuevo registro para una posición (p.ej 5) de la tabla

  tablaAlumnos[5] = new TipoAlumno();

- Acceso a los campos de un registro de la tabla

  tablaAlumnos[5].Nota = 7.5

  // Establece la nota de alumno en 5º posición

Inicialización de una tabla de datos definidos por el usuario

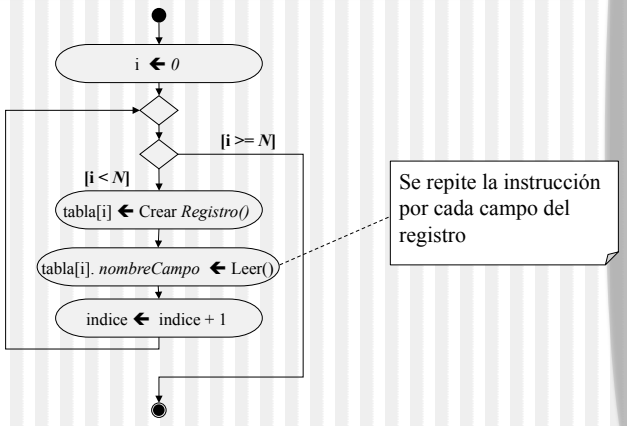


Tabla: operaciones

```
public static void inicializarTabla (TipoAlumno[] tabla) {
 BufferedReader ent = new BufferedReader(new InputStreamReader(System.in));
 try {
    for (int i=0; i < tabla.length;i++)
    {
            tabla[i]  = new TipoAlumno(); // Crear nuevo alumno
            System.out.println("Introduce el DNI del alumno");
            tabla[i].dni = ent.readLine();
            System.out.println("Introduce el nombre del alumno");
            tabla[i].nombre = ent.readLine();
            System.out.println("Introduce el apellido del alumno");
            tabla[i].apellido = ent.readLine();
    }
 }
 cacth (IOException ex) {}
}
```