

# CC1001-7 Auxiliar Extra – Control 3

## Arreglos, Computación Numérica y MATLAB

Daniel Calderon – Sebastián Fehlandt

3 de Junio de 2009

### P1) Búsqueda y Ordenamiento(30min)

a) Implemente una función de encabezado

```
public static int[] buscar(Comparable x, Comparable[][] y)
```

que retorne la posición {i,j} del elemento x en la matriz y. Suponga que la matriz esta ordenada de menor a mayor hacia la derecha, y luego hacia abajo.

b) Implemente una función de encabezado

```
public static int ordenarSegun(Comparable[] x, Comparable[] y)
```

que ordene “de menor a mayor” los arreglos x e y según las comparaciones realizadas con x. Y retorne el número de líneas de los arreglos. Utilícela en un programa que muestre una lista de pares (apellido\_alumno, notas), ordenándola alfabéticamente según el apellido del alumno.

c) (Propuesto) Implemente una función de encabezado

```
public static Object[][] revolver(int[] x, int[] y, Object[][] m)
```

que retorne un arreglo de Object donde el elemento i,j corresponda al elemento m[x[i]][y[j]].

### P3) Complejos y Matriz de Complejos (Propuesto)

Implemente la clase complejo con los siguientes métodos:

<code>private double real, imag;</code>	Partes real e imaginaria del complejo
<code>public Complejo(double x, double y)</code>	Constructor
<code>public double real()</code>	Retorna parte real del complejo
<code>public double imag()</code>	Retorna la parte imaginaria
<code>public double abs()</code>	Retorna el módulo del complejo
<code>public double ang()</code>	Retorna el ángulo del complejo
<code>public Complejo conjugado()</code>	Parte imaginaria cambia de signo
<code>public Complejo sumar(Complejo c)</code>	
<code>public Complejo restar(Complejo c)</code>	
<code>public Complejo multiplicar(double r)</code>	Retorna el producto por escalar del complejo
<code>public Complejo multiplicar(Complejo c)</code>	
<code>public Complejo invertir()</code>	$z^{-1} = \frac{\bar{z}}{ z ^2}$
<code>public Complejo dividir(Complejo c)</code>	

Implemente la Clase MatrizCompleja con los mismos métodos de la clase matriz pero definida por las variables de instancia:

```
protected Complejo[][] x; protected int n,m;
```

### P2) Clase Matriz (30min)

Implementa la clase Matriz con los siguientes métodos:

<code>protected double[][] x;</code> <code>protected int n,m;</code> <code>public Matriz(int n,int m)</code>	Variables de instancia Construye una matriz de nxm sin asignar valores
<code>public Matriz(int n,int m, double v)</code>	Construye una matriz de nxm asignando el valor v a todas las celdas
<code>public Matriz(double[][] x)</code>	Construye una matriz en base al arreglo x
<code>public int[] dim()</code> <code>public boolean equalsDim(Matriz w)</code>	Dimensión de la matriz. Ej: {n,m} True si tienen igual dimensión.
<code>public void set(double d,int i,int j)</code>	Asigna el valor “d” a la componente i,j de la Matriz
<code>public double get(int i,int j)</code>	Retorna la componente i,j de la Matriz
<code>public Matriz traspuesta()</code>	Retorna la matriz traspuesta
<code>public Matriz getSubmatriz(int i0,int j0,int i1,int j1)</code>	Entrega una submatriz formada desde la celda i0,j0 hasta la celda i1,j1
<code>public Matriz setSubmatriz(int i0,int j0, Matriz w)</code>	Modifica los valores de la submatriz a partir de i0,j0 poniendo los valores de la matriz w
<code>public Matriz ponderar(double a)</code>	Retorna una matriz ponderada en un factor a
<code>public Matriz sumar(Matriz w)</code>	Retorna la suma de ambas matrices
<code>public Matriz multiplicar(Matriz w)</code>	Retorna la matriz producto de ambas
<code>public void print()</code>	Imprime la matriz, entre elementos de la fila use “\t” y entre filas use “\n”
<code>public static Matriz rand(int n,int m)</code>	Genera una matriz de nxm con números aleatorios entre 0 y 1
<code>public static Matriz rand(int n)</code>	Genera una matriz de nxn con números aleatorios entre 0 y 1
<code>public static Matriz id(int n)</code>	Genera la matriz identidad de dimensión n

#### P4) MATLAB (45min)

a) Escriba la función de encabezado `function out=u(t)` que retorna el resultado de aplicar la función escalón unitario a  $t$  (que puede ser un número o un vector). La función escalón unitario se define como:

$$u(t) = \begin{cases} 1 & \text{si } t \geq 0 \\ 0 & \text{si no} \end{cases}$$

b) Escriba la función de encabezado `function out=rect(t)` que retorna el resultado de aplicar la función `rect` a  $t$  (que puede ser un número o un vector). La función `rect` se define como:

$$rect(t) = \begin{cases} 1 & \text{si } t \in [-0.5, 0.5] \\ 0 & \text{si no} \end{cases}$$

c) Escriba la función `function out = derivar(t,x)` donde  $t$  y  $x$  corresponden a vectores de la misma longitud cuyos elementos se relacionan de la siguiente manera  $x(i) = f(t(i))$ , donde  $f$  es alguna función, es decir, los pares ordenados  $(t(i), x(i))$  corresponden a puntos descritos por la función  $f$  en el plano cartesiano. La función debe retornar para cada índice  $i$ , la función derivada de  $f$  evaluada en dicho punto, mediante una aproximación por secantes.

d) Escriba la función:

`function out = integrarTrapecios(t,x)` donde  $t$  y  $x$  corresponden a los mismos vectores que en la parte anterior. La función debe retornar para cada índice  $i$ , la función primitiva de  $f$  evaluada en dicho punto, es decir, para cada índice  $i$ , debe retornar el área bajo la curva entre el índice 0 y el índice  $i$ .

e) Escriba una función que reciba los mismo parámetros que las 2 anteriores y grafique (con `subplot`) la función, su derivada y su primitiva.

f) Escriba un programa que pida al usuario el nombre de una función (m-file), intervalo a considerar y número de puntos y realice los mismo gráficos que en la parte anterior.