UNIVERSITAT JAUME·I

# 1º Ingeniería Informática
# 1º Matemática Computacional

## EI1008/MT1008 Programación II

Práctica 1: Revisión de conceptos básicos de programación

### Ejercicio 1

Escribe un programa que lea un número entero, $n$, y escriba su factorial, $n!$. Un ejemplo de ejecución del programa es:

```
Introduce un número entero: 5
5! = 120
```

El factorial de 15 es 1307674368000. ¿Es ese el resultado que obtienes al ejecutar tu programa? Si no lo es, piensa (*sin modificar tu código*) a qué puede deberse el error producido.

### Ejercicio 2

Escribe un programa que lea un número entero, $n$, y escriba su doble factorial, $n!!$. El doble factorial de $n$ se define como:

$$n!! = \begin{cases} 1, & \text{si } n = 0 \text{ o } n = 1 \\ 2 \times 4 \times 6 \times \ldots \times (n-2) \times n, & \text{si } n \text{ es par} \\ 1 \times 3 \times 5 \times \ldots \times (n-2) \times n, & \text{si } n \text{ es impar} \end{cases}$$

Un ejemplo de ejecución del programa es:

```
Introduce un número entero: 10
10!! = 3840
```

Otro ejemplo de ejecución del programa es:

```
Introduce un número entero: 7
7!! = 105
```

### Ejercicio 3

Escribe un programa que lea un número entero y escriba "Es primo" o "No es primo", según corresponda. Un ejemplo de ejecución del programa es:

```
Introduce un número entero: 167
Es primo
```

Otro ejemplo de ejecución del programa es:

```
Introduce un número entero: 1003
No es primo
```

### Ejercicio 4

Escribe un programa que lea un número entero, $n$, y escriba todos los números primos menores que $n$. Un ejemplo de ejecución del programa es:

```
Introduce un número entero: 23
Los números primos menores que 23 son: 2 3 5 7 11 13 17 19
```

### Ejercicio 5

Escribe un programa que lea un número entero, $n$, y escriba su primorial, $n\#$. El primorial de $n$ se define como el producto de todos los números primos menores o iguales a $n$. Un ejemplo de ejecución del programa es:

```
Introduce un número entero: 11
11# = 2310
```

Otro ejemplo de ejecución del programa es:

```
Introduce un número entero: 15
15# = 30030
```

### Ejercicio 6

Escribe un programa que lea un número entero, $n$, y escriba el mayor número comprendido entre 1 y $n$ con más divisores. Un ejemplo de ejecución del programa es:

```
Introduce un número entero: 100
El número con más divisores es 96 (12 divisores)
```

### Ejercicio 7

Escribe un método estático, `esPrimo`, que tenga como parámetro un número entero y devuelva `true` cuando ese número sea primo y `false` cuando no lo sea.

Reescribe el programa del ejercicio 3 para que haga uso del método `esPrimo`.

### Ejercicio 8

Reescribe el programa del ejercicio 4 para que haga uso del método `esPrimo`.

### Ejercicio 9

Reescribe el programa del ejercicio 5 para que haga uso del método `esPrimo`.

### Ejercicio 10

Escribe un método estático, `contarDivisores`, que tenga como parámetro un número entero y devuelva como resultado la cantidad de divisores de ese número.

Reescribe el programa del ejercicio 6 para que haga uso del método `contarDivisores`.

En los siguientes ejercicios se pide la implementación de diferentes métodos estáticos. En todos ellos deberás escribir también un método `main` que haga uso de los métodos implementados para probar su correcto funcionamiento mediante una pequeña batería de pruebas. Por ejemplo, en el ejercicio 11 podrías utilizar un código similar al siguiente:

```
public static void main(String[] args) {
    prueba("r.j.", "verde", false);
    prueba("r.j.", "rojo", true);
    prueba("r.j.", "reja", true);
    prueba("r.j.", "reza", false);
    prueba("r.j.", "rrjj", false);
    prueba("r.j.", "puja", false);
}

public static boolean encaja(String patrón, String cadena) {
    ...
}

public static boolean esVocal(char c) {
    return "AEIOUaeiou".indexOf(c) >= 0;
}

public static void prueba(String patrón, String cadena, boolean esperado) {
    boolean resultado = encaja(patrón, cadena);
    System.out.format("Prueba con %s y %s --> %s: ", patrón, cadena, resultado);
    if (resultado == esperado)
        System.out.println("ok");
    else
        System.out.format("error (se esperaba %s)%n", esperado);
}
```

## Ejercicio 11

Escribe un método estático, `encaja`, que tenga como parámetros dos cadenas de caracteres, `patrón` y `cadena`, y devuelva `true` cuando la cadena encaje en el patrón y `false` en caso contrario. Una cadena encaja en un patrón cuando:

- tiene la misma longitud,
- en las posiciones del patrón en las que aparece el carácter '.' hay una vocal en la cadena, y
- en las posiciones del patrón en las que hay un carácter distinto de '.' aparece exactamente el mismo carácter en la cadena.

Considera que ya existe[1] definido un método, `esVocal`, que recibe un carácter y devuelve `true` cuando el carácter se corresponde con una vocal y `false` en caso contrario.

## Ejercicio 12

Escribe un método estático, `contarPalabras`, que tenga como parámetro una cadena y devuelva la cantidad de palabras que aparecen en la misma. Consideraremos que una palabra está formada por una

[1] En el cuadro superior de la página tienes una posible implementación de este método en el que, por simplicidad, suponemos que no habrá vocales con tildes o diéresis.

secuencia de caracteres distintos del espacio en blanco. Ten en cuenta que pueden existir varios espacios entre dos palabras sucesivas, así como al comienzo y al final de la cadena. Los únicos métodos de cadenas que puedes usar son `length` y `charAt`.

### Ejercicio 13

Escribe un método estático, `obtenerPalabra`, que tenga como parámetros una cadena y un número entero `i` y devuelva la `i`-ésima palabra de la cadena cuando el valor de `i` sea válido (es decir, esté comprendido entre 1 y el número de palabras de la cadena) y `null` cuando no lo sea. Los únicos métodos de cadenas que puedes usar son `length`, `charAt` y `substring`.

### Ejercicio 14

Escribe un método estático, `últimaPosición`, que tenga como parámetros un vector de enteros y un número entero y devuelva la última posición del vector que contenga el elemento dado o un valor negativo cuando el elemento no aparezca en el vector.

### Ejercicio 15

Escribe un método estático, `estáOrdenado`, que tenga como parámetro un vector de cadenas y devuelva `true` cuando el vector esté ordenado lexicográficamente de menor a mayor y `false` cuando no lo esté.

### Ejercicio 16

Escribe un método estático, `contarOlasDeFrío`, que tenga como parámetros un vector de números de tipo `double` y un número entero `n`. Los valores del vector dado representan las temperaturas máximas de cada día en una ciudad a lo largo de un año. El método debe devolver la cantidad de olas de frío que sufrió la ciudad en ese año. Se considera que hay una ola de frío cuando la temperatura máxima está por debajo de 0 grados durante más de `n` días consecutivos.

### Ejercicio 17

Escribe un método estático, `hayRepetidos`, que tenga como parámetro un vector de enteros y devuelva `true` cuando en el vector haya elementos repetidos y `false` en caso contrario.

### Ejercicio 18

Escribe un método estático, `encajan`, que tenga como parámetros un patrón y un vector de cadenas y devuelva como resultado un nuevo vector de cadenas que contenga únicamente aquellas cadenas del vector dado que encajen en el patrón, tal y como se ha definido en el ejercicio 11. Por lo tanto, tu solución debe hacer uso de dicho método.

### Ejercicio 19

Escribe un método estático, `eliminarPosición`, que tenga como parámetros un vector de enteros y un entero que representa una posición en el vector y devuelva un nuevo vector con los mismos elementos que el vector dado, pero en el que se haya eliminado el elemento que ocupaba la posición indicada. Si la posición indicada no fuera válida, el método debe devolver el *mismo vector* que ha recibido como parámetro.

### Ejercicio 20

Escribe un método estático, `eliminarValor`, que tenga como parámetros un vector de enteros y un número entero y devuelva un nuevo vector que contenga los mismos elementos que el vector recibido, salvo aquellos que coincidan con el número dado.

## Ejercicio 21

Escribe un método estático, `contiene`, que tenga como parámetros un vector de enteros y un número entero y devuelva `true` cuando el número aparezca en el vector y `false` en caso contrario.

Haciendo uso de ese método, escribe otro método estático, `contiene`, que tenga como parámetros dos vectores de enteros (supón que no contienen elementos repetidos) y devuelva `true` cuando todos los elementos del segundo vector aparezcan en el primero y `false` en caso contrario.

## Ejercicio 22

Escribe un método estático, `posiciónInserción`, que tenga como parámetros un vector de enteros ordenado de menor a mayor y un valor entero y devuelva la posición en la que debería insertarse el valor dado para que el vector siguiese estando ordenado. Si el valor dado ya estuviera en el vector, se debería devolver la primera posición en la que se encontrase.

## Ejercicio 23

Escribe un método estático, `másOlasDeFrío`, que tenga como parámetros una matriz de números de tipo `double` y un entero `n`. Al igual que en el ejercicio 16, los valores de cada fila de la matriz representan las temperaturas máximas de cada día en una ciudad a lo largo de un año y se considera que hay una ola de frío cuando la temperatura máxima es negativa durante más de `n` días consecutivos. En este caso, cada fila de la matriz almacena las temperaturas de diferentes años, comenzando en 1900. Por simplicidad, supón que todos los años tienen el mismo número de días. El método debe devolver como resultado el año más reciente en el que se produjo un mayor número de olas de frío. Si no se hubiese producido ninguna ola de frío, se debería devolver un número negativo.

Para solucionar este ejercicio, debes emplear el método implementado en el ejercicio 16.

## Ejercicio 24

*Alquimia* es un juego que consiste en obtener nuevos elementos a partir de cuatro elementos básicos (agua, aire, fuego y tierra) mediante determinadas combinaciones de parejas de elementos. El vector `elementos` contiene los nombres de todos los elementos posibles, incluidos los cuatro elementos básicos. Los nombres de los elementos no contienen espacios en blanco. Las combinaciones permitidas se almacenan en un fichero de texto. Cada línea de este fichero representa una combinación y contiene, en este orden, los nombres de dos elementos combinables y del elemento que generan. Un ejemplo del contenido del fichero sería:

```
tierra   fuego  lava
lava     aire   piedra
piedra   agua   arena
piedra   aire   arena
arena    arena  desierto
arena    fuego  cristal
cristal  arena  reloj_de_arena
```

Escribe un método estático, `crearMatrizCombinaciones`, que tenga como parámetros un vector de elementos y el nombre de un fichero de combinaciones. El método debe devolver como resultado una matriz simétrica de $N \times N$ enteros (siendo $N$ el número de elementos), donde cada celda represente el resultado de la combinación de una pareja de elementos. Las celdas `[i][j]` y `[j][i]` deben almacenar el entero `k` si al combinar, en cualquier orden, `elementos[i]` y `elementos[j]` se obtiene `elementos[k]`. Si los elementos no son combinables, se almacenará el valor `-1`.

## Ejercicio 25

Partiendo del código que has implementado en el ejercicio anterior, añade un nuevo método estático, `elementoGenerado`, que tenga como parámetros una matriz de combinaciones, un vector de elementos y

dos cadenas. Si las cadenas se corresponden con dos elementos que pueden ser combinados, el método debe devolver como resultado el nombre del nuevo elemento obtenido. Si no, debe devolver `null`.

### Ejercicio 26

Partiendo del código que has implementado en el ejercicio anterior, añade un nuevo método estático, `combinablesConsigo`, que tenga como parámetros una matriz de combinaciones y un vector de elementos y devuelva como resultado un nuevo vector con los nombres de todos los elementos que pueden ser combinados consigo mismos.

### Ejercicio 27

Partiendo del código que has implementado en el ejercicio anterior, añade un nuevo método estático, `contarTerminales`, que tenga como parámetro una matriz de combinaciones y devuelva como resultado la cantidad de elementos terminales.

Un elemento es *terminal* cuando no es posible generar nuevos elementos a partir de él, es decir, cuando todas las celdas correspondientes a su fila (o columna) en la matriz de combinaciones contienen el valor -1.

### Ejercicio 28

Partiendo del código que has implementado en el ejercicio anterior, añade un nuevo método estático, `elementoMásCombinable`, que tenga como parámetros una matriz de combinaciones y un vector de elementos y devuelva como resultado el nombre del elemento más combinable. En caso de empate entre varios elementos, el método podrá devolver cualquiera de ellos.

El elemento *más combinable* es aquel que puede ser combinado con un mayor número de elementos, es decir, aquel que en su fila (o columna) de la matriz de combinaciones tiene un mayor número de celdas diferentes de `-1`.

### Ejercicio 29

Partiendo del código que has implementado en el ejercicio anterior, añade un nuevo método estático, `múltiplesCombinaciones`, que tenga como parámetro una matriz de combinaciones y devuelva como resultado la cantidad de elementos que pueden obtenerse a partir de múltiples parejas de elementos combinables.

Por ejemplo, el único modo de obtener lava sería combinando tierra y fuego. En cambio, para obtener arena podríamos combinar tanto piedra y agua como aire y piedra.