Actividad 12

Ejercicios funciones, bucles, cadenas y vectores.

Alumno: Herrera Flores Barbara Isabel

Profesor: José Javier Faro Rodríguez

Grupo: DAW

UF02 Diseño modular

Índice

Ejercicio 1

Fer una funció static int major2 (int a, int b) a la que passarem dos valors enters i retornarà el major d'ells.

```
static int major2(int a, int b) {
    if (a > b)
        return a;
    return b;
}
```

Ejercicio 2

Fer una funció static int major3 (int a, int b, int c) a la que passarem tres valors enters i retornarà el major valor entre ells

```
static int major3(int a, int b, int c) {
    return major2(major2(a, b), c);
}
```

Ejercicio 3

Fer una funció static int major4 (int a, int b, int c, int d) a la que passarem quatre valors enters i retornarà el major d'ells

```
static int major4(int a, int b, int c, int d) {
    return major2(major2(a, b), major2(c, d));
}
```

Ejercicio 4

Programar la funció int quantesVegades (String cad, char c) que ens retorna el nombre de vegades que apareix el caràcter c a la cadena cad.

```
static int quantesVegades(String cad, char c) {
    int contador = 0;
    for (int i = 0; i < cad.length(); i++)
        if (cad.charAt(i) == c)
            contador++;
    return contador;
}
```

Ejercicio 5

Programar una funció int quantsEspais(String cad) a la que passarem una cadena cad i ens retornarà el nombre d'espais que conté.

```
static int quantsEspais(String cad) {
    return quantesVegades(cad, ' ');
}
```

Ejercicio 6

Programar una funció int quantesVocals(String cad, String sonVocals) a la que passarem una cadena cad i una altra sonVocals contenint totes les vocals que volem considerar dins de cad i retornarà el nombre de vocals (de les contingudes a sonVocals) que hi ha dintre de cad.

```
static int quantesVocals(String cad, String sonVocals) {
    int contador = 0;
    for (int i = 0; i < sonVocals.length(); i++) {
        quantesVegades(cad, sonVocals.charAt(i));
        contador++;
    }
    return contador;
}
```

Ejercicio 7

Programeu la funció String separa(String cad, String separador), a la que passarem una cadena cad i una altra contenint un caràcter (o més) de separació i retornarà una altra cadena on ha separat cadascú dels caràcters de cad amb el separador indicat.

```
static String separa(String cad, String separador) {
    String content = "";
    for (int i = 0; i < cad.length(); i++) {
        content += cad.charAt(i) + separador;
    }
    return content.substring(0, content.length() - separador.length());
}
```

Ejercicio 8

Fer una funció hiHaBruixa(cad) que ens retorni true si dintre de cad hi ha alguna bruixa (al menys una) i false en cas contrari.

```
static boolean hiHaBruixa(String cad) {
    return cad.contains("=()=");
}
```

Ejercicio 9

Fer una funció String explanada(int n) que retornarà una cadena de longitud n generada de forma aleatòria i contenint únicament els caràcters '(', ')' i '='.

```
static String explanada(int n) {
    String cadena = "()=";
    String aleatorio = "";
    for (int i = 0; i < n; i++)
        aleatorio += cadena.charAt((int) (Math.random() * cadena.length()));
    return aleatorio;
}
```

Ejercicio 10

A partir de les dues funcions anteriors, feu una funció explanadaBruixa(n), que retorni una cadena de longitud n (amb n major o igual a 10 i menor de 100) que contingui almenys una bruixa.

```
static String explanadaBruixa(int n) {
    String contenido = "";
    if (n < 10 || n > 100) {
        return "";
    }
    while (!hiHaBruixa(contenido)) {
        contenido = explanada(n);
    }
    return contenido;
}
```

Ejercicio 11

Feu una funció quantesBruixes(explanada), a la que passarem una cadena explanada i retornarà quantes bruixes hi ha en aquesta, es a dir, quantes vegades està repetida la combinació “=()=” dintre de la cadena explanada.

```
static int quantesBruixes(String explanada) {
    int cuantas = 0;
    for (int n = 0; n <= explanada.length() - 4; n++) {
        if (explanada.substring(n, n + 4).equals("=()=")) {
            cuantas++;
            n += 3;
        }
    }
    return cuantas;
}
```

Ejercicio 12

Programeu una funció explanadaBruixes(n,numBruixes), a la que passarem dos enters i retornarà una cadena de longitud n (amb n major o igual a 50 i menor de 1000) que contingui al menys num_bruixes bruixes.

```
static String explanadaBruixes(int n, int numBuixes) {
    String cadena = "";
    if (n >= 50 && n < 1000 && numBuixes < 5) {
        do {
            cadena = explanada(n);
        } while ((quantesBruixes(cadena) < numBuixes));
        return cadena;
    }
    return "";
}
```

Ejercicio 13

Programeu una funció static String mataBruixa(String explanada), a la que passarem una explanada contenint com a mínim una bruixa (la funció no ho comprovarà, assumirà que a la explanada hi ha com a mínim una bruixa) i retornarà una cadena on ha eliminat la primera bruixa que trobi a l’explanada (des de l’esquerra).

```
static String mataBruixa(String explanada) {
    int bruixa = explanada.indexOf("=()=");
```

```
    return explanada.substring(0, bruixa) + explanada.substring(bruixa + 4);
}
```

Ejercicio 14

Programar una funció static int mataBruixes(String explanada) a la que passem una cadena explanada on pot haver-hi un nombre indeterminat de bruixes, les mati (tregui els caràcters "=()=" de la cadena) i finalment retorni el número d'elles que ha eliminat.

```
static int mataBruixes(String explanada) {
    int cuantas = 0;
    while (hiHaBruixa(explanada)) {
        explanada = mataBruixa(explanada);
        cuantas++;
    }
    return cuantas;
}
```

Ejercicio 15

Feu una funció static int max(int v[]) a la que passareu un vector d'enters i retornarà el major valor d'aquest vector.

Solució

Trobar el major índex d'un array utilitzant el mètode max_sort és més lent, tenint en compte que cal trucar a un altre mètode (sort) per organitzar, a més generar una còpia de l'array i finalment retornar l'últim element de l'array, aquest mètode triga 01:00 aproximat de 1.081.600 nano segons, molt més si ho comparem amb el mètode max, que triga aproximadament 445.800 nano segons.

```
static int max_sort(int v[]) {
    int[] value = new int[v.length];
    System.arraycopy(v, 0, value, 0, v.length);
    Arrays.sort(value);
    return value[value.length - 1];
}

static int max(int v[]) {
    int maxValue = v[0];
    for (int value : v) {
        if (value > maxValue)
            maxValue = value;
    }
    return maxValue;
}
```

Método main:

```
    public static void main(String[] args) {
        long tiempoNs = System.nanoTime();
        int ejemplo[] = {10000, 22, 848485, 5, 5, 1};
        System.out.println(max_sort(ejemplo));
        System.out.println("Tiempo invertido = " + (System.nanoTime() -
tiempoNs));
```

```
    }
}
```

Ejercicio 16

Sobrecarregueu la funció de l'exercici anterior, es a dir, creeu una altra funció que declarareu com double max(double v[]) a la que passareu un vector d'elements de tipus double i retorna el major d'ells.

```
static double max(double v[]) {
    double[] value = new double[v.length];
    System.arraycopy(v, 0, value, 0, v.length);
    Arrays.sort(value);
    double max = value[value.length - 1];
    return max;
}
```

```
static double max_sort(double v[]) {
    double[] value = new double[v.length];
    System.arraycopy(v, 0, value, 0, v.length);
    Arrays.sort(value);
    return value[value.length - 1];
}
```

Ejercicio 17

Programar una funció min(vector) a la que passarem un vector de tipus int o long (l'haureu de sobrecarregar) i ens retornarà l'element de valor mínim

```
static int min(int v[]) {
    int[] value = new int[v.length];
    System.arraycopy(v, 0, value, 0, v.length);
    Arrays.sort(value);
    int min = value[0];
    return min;
}
```

Ejercicio 18

Programar una funció suma(vector) a la que passarem un vector de números reals (de tipus double) i ens retornarà la suma de tots els seus valors.

```
static double suma(double vector[]) {
    double sum = 0;
    for (int i = 0; i < vector.length; i++) {
        sum += vector[i];
    }
    return sum;
}
```

Ejercicio 19

Programar una funció mitja(vector) a la que passarem un vector de valors de tipus long i ens retornarà el valor mig (de tipus double), que s'obté sumant tots els elements i dividint-los entre el número d'elements del vector. Suposem que tots els elements del vector contenen dades vàlides, es a dir, que està ple.

```
Sobrecarga de metodos
//Ejercicio 19
static double suma(long vector[]) {
    double sum = 0;
    for (int i = 0; i < vector.length; i++) {
        sum += vector[i];
    }
    return sum;
}

static double mitja(long[] vector) {
    return suma(vector) / vector.length;
}
```

Ejercicio 20

Inicialitzeu aleatòriament un vector d'un milió d'elements de tipus double amb valors majors o iguals a 0 i menors que 1 (amb Math.random()), i comproveu quin dels dos mètodes comentats a l'exercici 15 és més ràpid per obtenir el valor màxim d'aquest vector. Consulteu la teoria Vectors i Matrius en Java i Temps d'execució en java per saber com fer-ho.

```
double[] vecMillion = new double[1000000];
for (int n = 0; n < vecMillion.length; n++)
    vecMillion[n] = (int) (Math.random() * 2);
long tiempoMs = System.currentTimeMillis();
double max = max(vecMillion);
System.out.println("Max()= " + max + " Tiempo invertido " +
(System.currentTimeMillis() - tiempoMs));
tiempoMs = System.currentTimeMillis();
max = max_sort(vecMillion);
System.out.println("Max_Sort()= " + max + " Tiempo invertido " +
(System.currentTimeMillis() - tiempoMs));
```

Com es va esmentar en l'exercici 15, el mètode max () és més ràpid, no ha de trucar a un altre mètode ni crear una còpia de l'array.

```
"C:\Program Files\Java\jdk-13.0.1\bin\java.exe" "
Max()= 1.0 Tiempo invertido 9
Max_Sort()= 1.0 Tiempo invertido 105
```

Ejercicio 21

Programeu una funció static int[] gira_vector(int[] v) a la que passarem un vector d'enters i li donarà la volta, posant en primer lloc el últim element del vector i així successivament.

```java
static int[] gira_vector(int[] vec) {
    int n, m, aux = 0;
    for (n = 0, m = vec.length - 1; n < vec.length / 2; n++, m--) {
        aux = vec[n];
        vec[n] = vec[m];
        vec[m] = aux;
    }
    return vec;
}
```

Ejercicio 22

Fer una funció static int[] suma_vectors(int[] v1, int[] v2) que retornarà un vector que representa la suma dels elements de dos vectors v1 i v2.

```java
public static int[] suma_vectors(int[] v1, int[] v2) {
    if (v1.length > v2.length) {
        int aux[] = v1;
        v1 = v2;
        v2 = aux;
    }
    int min = v1.length;
    int max = v2.length;
    int[] nuevo = new int[max];
    for (int i = 0; i < min; i++) {
        nuevo[i] = v1[i] + v2[i];
        System.arraycopy(v2, min, nuevo, min, max - min);
    }
    return nuevo;
}
```

Ejercicio 23

Programeu una funció static int[] crea_vector (int min, int max, int num_valors) que retorna un vector de num_valors de tipus int compresos entre min i max (inclosos) i generats aleatòriament. Si min > max, s'intercanviaran.

```java
static int[] crea_vector(int min, int max, int num_valors) {
    int[] vector = new int[num_valors];
    if (min > max) {
        int aux = min;
        min = max;
        max = aux;
    }
    for (int n = 0; n < vector.length; n++) {
        vector[n] = (int) (Math.random() * (max - min + 1) + min);
    }
    return vector;
}
```

Ejercicio 24

Fer una funció static int posició(int [] v, int x) que retornarà la posició del valor x (la primera aparició d'aquest valor) dintre del vector v. Si el valor x no apareix dintre del vector v, la funció haurà de retornar el valor -1.

```java
static int posicio(int[] v, int x) {
    for (int i = 0; i < v.length; i++)
        if (v[i] == v[x])
            return v[i];
    return -1;

}
```

Ejercicio 25

Fer una funció static int quants(int [] v, int x) que retornarà quantes vegades apareix el valor x dintre del vector v

```java
static int quants(int[] v, int x) {
    int contador = 0;
    for (int i = 0; i < v.length; i++)
        if (v[i] == x)
            contador++;
    return contador;

}
```

Ejercicio 26

Fer una funció static long [] vec_primers(int num_primers) que retorna un vector amb els primers num_primers que son primers (que no son divisibles més que per ells mateixos i per la unitat).

```java
static boolean num_primers(long n) {
    if (n <= 2)
        return n == 2;
    if (n % 2 == 0)
        return false;
    long sq = (long) Math.sqrt(n);
    for (int m = 3; m <= sq; m += 2)
        if (n % m == 0)
            return false;
    return true;
}

static long[] vec_primers(int num_primers) {
    long[] primos = new long[num_primers];
    primos[0] = 2;
    long provar = 3;
    int num_primos = 1;
    while (provar <= num_primers) {
        if (num_primers(provar))
            primos[num_primos++] = provar;
        provar += 2;
    }
    return Arrays.copyOf(primos, num_primos);
}
```

Ejercicio 27

Programeu una funció static void inserta(int [] v, int x, int p) que insereix al vector v el valor x a la posició p (desplaçament sobre el vector).

```
static void inserta(int[] v, int x, int p) {
    int[] copyV = Arrays.copyOf(v, v.length);
    if (0 == p) {
        v[0] = x;
        for (int i = 0; i < v.length - 1; i++)
            v[i + 1] = copyV[i];
    } else if (v.length - 1 == p) {
        v[v.length - 1] = x;
    } else {
        for (int i = p; i < v.length; i++) {
            v[p] = x;
            v[i] = copyV[i - 1];
        }
    }
    System.out.println(Arrays.toString(v));
}
```

Ejercicio 28

Programeu una funció static boolean conte(int[] v, int[] v_dins) que retornarà true si el vector v conté el vector v_dins, és a dir, si tots els elements de v_dins estan consecutius en v.

Solució

Amb el primer bucle recorro l'array on es buscarà el contingut de l'altre, amb continuï obtinc la primera posició de l'vector amb el qual compararé, a partir d'aquesta posició comença a incrementar-se per cada valor que retorni true, en cas contrari la variable auxiliar retorna false ja "p" se li resta un element perquè començarà una altra vegada el bucle principal (tots dos bucles incrementen p).

```
static boolean conte(int[] v, int[] v_dins) {
    boolean aux = false;
    for (int p = 0; p < v.length; p++) {
        if (v[p] != v_dins[0]) {
            continue;
        }
        for (int j = 0; j < v_dins.length; j++, p++) {
            if (v[p] == v_dins[j]) {
                aux = true;
            } else {
                aux = false;
                p--;
                break;
            }
        }
    }
    return aux;
}
```