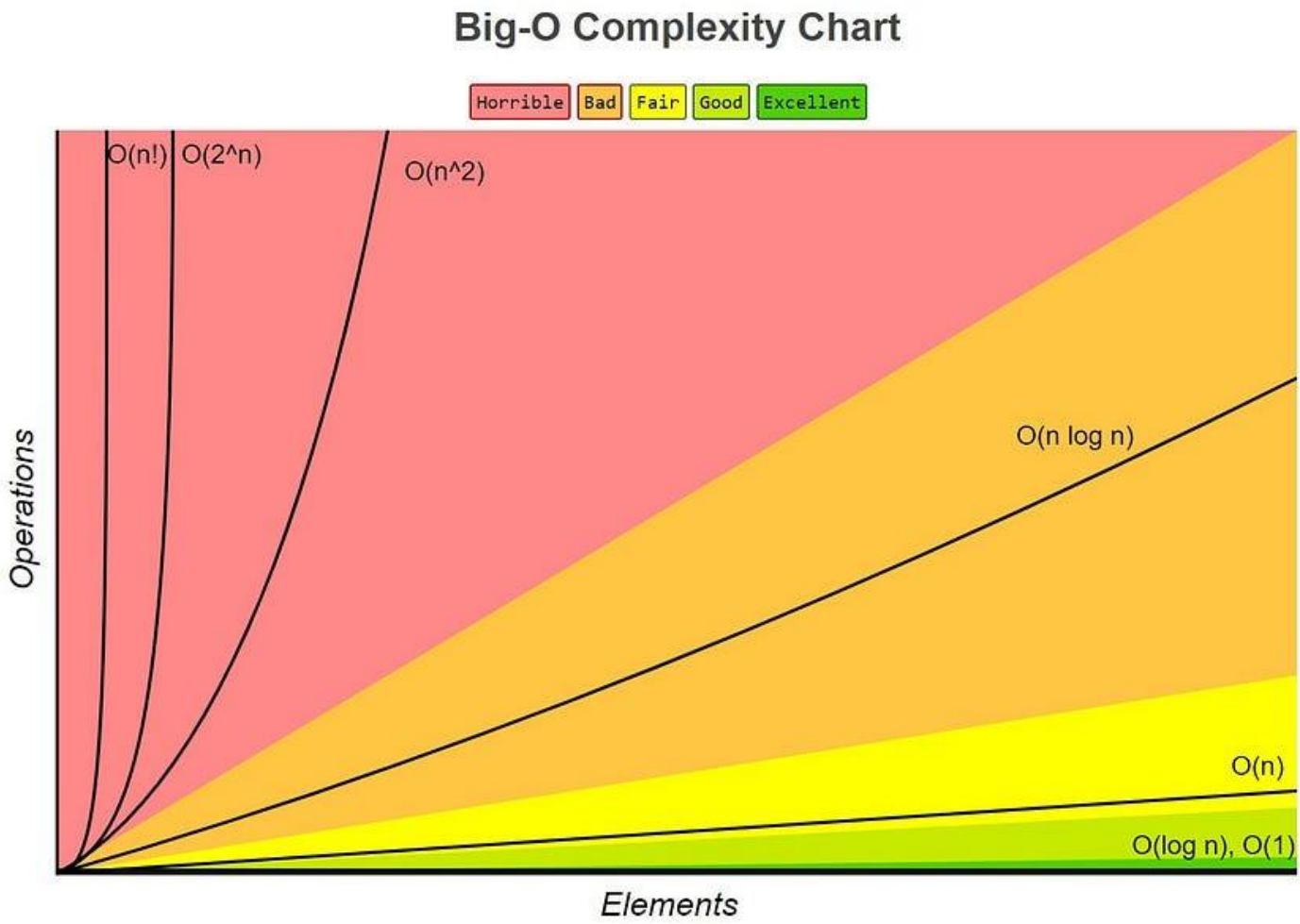
FUNDAMENTOS DE LA
PROGRAMACIÓN

2º CUATRIMESTRE DTIE MATEMÁTICAS Y FÍSICA – CURSO 2024/25.



TEMA 1: RESOLUCIÓN DE PROBLEMAS Y ALGORITMOS

Dos fases:

1. Del enunciado al algoritmo
2. Implementar el algoritmo

Enunciado → Algoritmo → Programa

Algoritmo: conjunto de instrucciones bien definidas, ordenadas y finitas. La estudia la algoritmia.

- Usar el infinitivo
- Si X entonces Y
- Mientras X, hacer Y ó Hacer N-veces X
- Guardar en variables

Técnicas de diseño

1. Top-Down (divide y vencerás): descomponer en subproblemas
2. Stepwise refinement (pasito a pasito): ir poco a poco, puede no ser lo más óptimo.
3. Backtracking: probar todo.

TEMA 2: INTRODUCCIÓN A LA PROGRAMACIÓN

Codificación (escribir) → Compilar → Ejecutar → Probar → Depurar (si no funciona)

Lenguajes de programación

1. Según su abstracción: Máquina → Ensamblador → Alto nivel
2. Según su traducción: Compilados* ó Interpretados
3. Según su tipado: Fuerte* ó Débil

Introducción al processing

IMPORTANTE: NO ES LO MISMO HACER ++variable QUE variable++, variable++ PRIMERO MANDA EL VALOR DE VARIABLE Y POSTERIORMENTE INCREMENTA UNO (POST-INCREMENTO) MIENTRAS QUE ++variable PRIEMRO INCREMENTA UNA UNIDAD Y LUEGO ENVÍA LA VARIABLE (PRE-INCREMENTO)

Operación	Aritmética	Programación
Suma	$+$	$+$
Resta	$-$	$-$
Multiplicación	$\times$	$*$
División	$\div$	$/$
Resto	mod	%

Números	Aritmética	Programación
Enteros	$\ldots, -2, -1, 0, 1, 2, \ldots$	$\ldots, -2, -1, 0, 1, 2, \ldots$
Decimales	, (coma)	. (punto)
Reales	π	3.1415 Una aproximación. Punto flotante.

Processing	Matemáticas	Significado
$abs(x)$	$\lvert x \rvert$	Valor absoluto
$ceil(x)$	$\lceil x \rceil$	Entero más cercano superior
$exp(x)$	e^x	Exponencial en base e
$floor(x)$	$\lfloor x \rfloor$	Entero más cercano inferior
$log(x)$	$\ln(x)$	Logaritmo en base e
$pow(x, y)$	x^y	Potencia y en base x
$round(x)$	$x \approx x'$	Redondeo de x es x'
$sq(x)$	x^2	Cuadrado de x
$sqrt(x)$	$\sqrt{x}$	Raíz cuadrada de x

TEMA 3: VARIABLES

Una variable es un espacio en la memoria dedicado a almacenar un dato

1 byte = 8 bits, bit = 0,1

No todos los números reales se pueden codificar (porque entre dos reales hay infinitos reales, es decir necesitaríamos infinita memoria para codificarlos). Solamente alcanzamos cierta precisión.

- Variable: lowerCamelCaseNotation (¡Empezar solo por letras minúsculas o "_" y no usar las palabras reservadas del lenguaje!), usar variables LOCALES.
- Constante: CAPSNOTATION → *final tipo_de_dato NOMBRE = valor;*

Tipos de variables: Para convertir entre tipos: *tipo1 ejemplo = valor; tipo 2 = (tipo2) ejemplo;* (Casting de tipos)

1. Simples
    a. Numéricas
        i. *byte* [-128, 127]
        ii. *int* (entero)*
        iii. *float* (real)*
        iv. *long* (entero más largo)
        v. *double* (real con más precisión)
    b. Caractéres: comillas simples (2 byte) → *char ejemplo = 'x' ;*
    c. Booleanos: {true, false} (1 byte) → *boolean ejemplo = false;*
2. Complejas: mezcla de simples
        i. *String* (cadenas de caracteres) → *String ejemplo = "¡Feliz año!";*
        ii. *arrays, registros* (más adelante)

Además podemos distinguir entre variables de entrada (dado...), variables intermedias (auxiliares) y variables de salida (lo que podemos retornar).

```
/* PROBLEMA 3.11

Escribe un programa que genere un número aleatorio en cierto intervalo [min,max] con 3 números decimales.

*/

int decimal1 = (int) random(0,9); // Incluye el extremo izquierdo pero NO el derecho

int decimal2 = (int) random(0,9);

int decimal3 = (int) random(0,9);


// Vamos a suponer que los extremos del intervalo han de ser enteros

int maximo = (int) random(-100, 100);

int minimo = (int) random(-100, maximo);


int numero = (int) random(minimo, maximo+1);


println("Generando número en el intervalo [" + minimo + ", " + maximo + "]..."); // Ejemplo de concatenación

println("El número generado es: " + numero + "," + decimal1 + decimal2 + decimal3);
```

```
/* PROBLEMA CLASE 2

Dada una cantidad de dinero en euros, escribe un programa que lo convierta a dólares americanos. Redondear el resultado a tres decimales.

*/

float euros = random(0, 100);

float dolares; // sin tilde

float factorEuroDolar = 1.04; // 1€= 1.04$ a día 28/01/2025


dolares = 150 * factorEuroDolar;


int cifrasDecimales = 3;

dolares = dolares * pow(10, 3) // Multiplicamos por mil

dolares = (int)dolares // Truncamos a entero

dolares = dolares / pow(10, cifrasDecimales);


println(“Euros: ” + euros + “, Dólares: ” + dolares);
```

```
/* PROBLEMA 3.16

Escribe un programa que intercambie los valores de 3 variable. Muestra su contenido antes y despúes del intercambio.


A --> B --> C --> A (intercambio circular)


SOLO SE PERMITE EL USO DE UNA ÚNICA VARIABLE AUXILIAR

*/

int var1 = (int)random(0,100);

int var2 = (int)random(0,100);

int var3 = (int)random(0,100);


println("1| Variable 1 = " + var1 + " Variable 2 = " + var2 + “ Variable 3 = “ + var3);


int aux = var2;

var2 = var1; // A --> B

var1 = var3; // C --> A

var3 = aux;  // B --> C


println("2| Variable 1 = " + var1 + " Variable 2 = " + var2 + “ Variable 3 = “ + var3);
```

TEMA 4: PROGRAMACIÓN ESTRUCTURADA

Algoritmo para dados dos números, sumar los pares y multiplicar los impares que hay entre ambos.

1. Almacenar en n y m (n<m) valores dados por el usuario.
2. Asignar actual igual a n.
3. Asignar suma igual a 0.
4. Asignar producto igual a 0.
5. Mientras que actual sea menor o igual que m, hacer:
    a. Imprimir el valor actual.
    b. Si actual es par, hacer:
        i. Sumar actual con suma para actualizar suma
    c. Si actual es impar, hacer:
        i. Multiplicar actual con producto para actualizar producto
6. Imprimir suma y producto).

Expresiones booleanas (al evaluarse dan un booleano).

1. Operadores condicionales (para variables numéricas):
    a. Igualdad → *a==b*
    b. Diferencia (a distinto de b) → *a != b* Es lo mismo a *!(a == b)*
    c. Desigualdad → *a>b, b<a, a>=b, b>=a*
2. Operadores booleanos:
    a. NOT →*!a*
    b. AND →*a&&b*
    c. OR →*a || b*
    d. XOR → *a^b*

Por ejemplo, lo que en matemáticas es *a=b=c*, en programación es *(a==b) && (b== c)*.

Condicional if

if (condición 1) {

instrucciones (si se cumple la condición 1)

} else if (condición 2) {

Instrucciones (si NO se cumple la condición 1 pero SÍ se cumple la condición 2)

} else {

Instrucciones (si no se cumple ninguna de las condiciones anteriores)

}

Instrucciones fuera del condicional (se ejecutará siempre)

```
/*PROBLEMA 4.19

Haz un programa que determine si un número x está en un intervalo [a,b]

*/

float x = random(-100, 100);

float minimo = random(-50, 50);

float maximo = random(minimo, 50);


println("x = " + x);

println("El intervalo es: [" + minimo + ", " + maximo + "]");


if ( (minimo <= x) && (x <= maximo) ) {println("x está dentro!");} else {println("x está fuera...");}
```

```
/*PROBLEMA 4.11
Dado el peso, altura y género calcular el IMC y clasificar.
*/
float peso = random(20, 150);
float altura = random(1.20, 2.15);
boolean genero;

// ASIGNAR UN VALOR ALEATORIO A UN BOOLEANO
if (random(0,1) < 0.5) {
          genero = true;
          println(“Es un hombre”);
} else {
          genero = false
          println(“Es una mujer”);
}

float IMC = (peso / pow(altura, 2));
println("La altura es: " + altura + " y el peso es: " + peso);
println("El índice de masa corporal es: " + IMC);

if (IMC <= 18) {
  println("NIGERIA");
} else if (IMC <= 24) {
  println("HUMANO");
} else if (IMC <= 29) {
  println("SEBOSO");
} else if (IMC <= 39) {
  println("BIEN GORDICO");
} else {
  println("LA MADRE DEL MARCOS");
}
```

```
/*PROBLEMA 4.18
Haz un programa que determine si dos puntos coinciden en alguna coordenada.
Por simplicidad suponga que los puntos solo toman valores enteros.
*/
int x1 = (int)random(-10, 11); // porque el intervalo es [-10, 11)
int x2 = (int)random(-10, 11);
int y1 = (int)random(-10, 11);
int y2 = (int)random(-10, 11);

println("A=(" + x1 + ", " + y1 + "); B=(" + x2 + ", " + y2 + ")");
if ((x1 == x2) && (y1 == y2)) {println("Los puntos son iguales"); }
else if (x1 == x2) {println("La primera componente de los puntos coincide");}
else if (y1 == y2) {println("La segunda componente de los puntos coincide");}
else {println("Los puntos no coinciden");}
```

```
/* PROBLEMA 4.23
Dados dos circulos determinados por sus respectivos centros y radios determinar si se solapan o no
*/

float cx1 = random(-10, 10); // Componentes (x1, y1); (x2, y2) de los centros de los círculos
float cy1 = random(-10, 10);
float cx2 = random(-10, 10);
float cy2 = random(-20, 10);

println("C1=(" + cx1 + ", " + cy1 + "); C2=(" + cx2 + ", " + cy2 + ")");

float maxRad = 3; // Máximo número que puede alcanzar el radio de ambos círculos
float rad1 = random(0, maxRad);
float rad2 =  random(0,maxRad);

println("R1 = "+ rad1 +", R2 ="+ rad2);

float distanciaCentros = sqrt( sq(cx1 -cx2) + sq(cy1 - cy2) );

println("La distancia entre ambos círculos es: "+ distanciaCentros);

if (distanciaCentros <= (rad1 + rad2) ){
  println("Los círculos se solapan");
} else {
  println("Los círculos no se solapan");
}
```

```
/*PROBLEMA 4.12
Dado un número entre el 1 y el 12 indique el nombre del mes y el número de días que tiene dicho mes.

SOLO SE PERMITEN 3 PRINT PARA MOSTRAR EL NÚMERO DE DÍAS
*/

int n = (int)random(1,13);
println("Número del mes: " + n);

//Esto ejecuta el primer print
if (n==1) {println("El mes es enero");}
else if (n==2)  {println("El mes es febrero");}
else if (n==3)  {println("El mes es marzo");}
else if (n==4)  {println("El mes es abril");}
else if (n==5)  {println("El mes es mayo");}
else if (n==6)  {println("El mes es junio");}
else if (n==7)  {println("El mes es julio");}
else if (n==8)  {println("El mes es agosto");}
else if (n==9)  {println("El mes es septiembre");}
else if (n==10) {println("El mes es octubre");}
else if (n==11) {println("El mes es noviembre");}
else if (n==12) {println("El mes es diciembre");}

// Aquí se ejecuta el segundo print
if (n <= 7) {
  if (n == 2) {
    println("El mes tiene 28 días");
  }else if (n%2 == 1) {
    println("El mes tiene 31 días");
  } else {
    println("El mes tiene 30 días");
  }
} else {
  if (n%2 == 1) {
    println("El mes tiene 30 días");
  } else {
    println("El mes tiene 31 días");
  }
}
```

Otras estructuras de control del programa son:

Bucle while:

Repite un código todo el tiempo mientras se tiene una condición (Para cuando no sabemos cuantas iteraciones hay que hacer)

while (condición) {

Instrucciones (si se cumple la condición);

Modificar la condición (si no, tendríamos un bucle infinito);}

Bucle for:

Repite un código un número de veces determinado (Para cuando sabemos <<implícitamente>> cuantas iteraciones hay que hacer);

for (iniciamos variable contador; condición de la variable contador; cambio de la variable contador) {

Instrucciones (si se cumple la condición que le hemos impuesto a la variable contador);

}

En general como contador utilizamos *int i;*

Condicional switch:

switch (Variable a evaluar) {

case valor1:

Instrucciones a evaluar si se dá el valor 3 para la "variable a evaluar"

case valor2:

Instrucciones a evaluar si se dá el valor 3 para la "variable a evaluar"

case valor3:

Instrucciones a evaluar si se dá el valor 3 para la "variable a evaluar"

.....

default:

Instrucciones a evaluar si no se ejecuta ninguno de los case

}

```
/*PROBLEMA CLASE
Calcula 2^n para cierto n natural positivo empleando únicamente productos
*/
int n = (int) random(0,11); print("2^" + n + " = ");
int resultado = 1;
for (int i = 0; i < n; i++) {resultado *= 2;}
print(resultado);
```

```
/* PROBLEMA 4.13
Dado un natural s, sumar todos los naturales hasta que dicha suma
sea mayor o igual que s
*/
int S = (int)random(0,100);
int contador = 1;
int suma = 0;
println("El número es: " + S);

// La suma ha de ser menor estricta porque si fuera menor o igual puede quedarse "a un paso" de completarse.
while (suma < S) {
  suma+=contador;
  contador++;
}

println("La suma es: " + suma);
```

```
/*` PROBLEMA 4.15
Mostrar todos los enteros en el intervalo [a, b] en orden inverso
*/

int min = (int)random(0, 10.1);
int max = (int)random(min, 50.1);

println("El intervalo es: [" + min + ", " + max+"]");

for (int i = max; i>= min; i--) {
  println(i);
}
```

```
/*PROBLEMA 4.26
Contar el número de divisores que tiene un número natural
*/
int numero = (int) random(1, 1000);
println("El número es: " + numero);
int numeroDeDivisores = 0;

for (int i = 1; i <= numero; i++) {
  if (numero % i == 0){
         println(i);
         numeroDeDivisores++;
         }
}

Println("EL NÚMERO DE DIVISORES ES: + " numeroDeDivisores);
```

```
/*PROBLEMA 4.14
Un número perfecto es un número entero positivo que es igual a la suma de sus divisores propios positivos.
Haz un programa que determine si un número es o no perfecto
*/

int numero = (int)random(0,100.1);
println("El número es: " + numero);

int sumaDivisores = 0;
// Tiene que ser estricto porque estamos contando los divisores propios
for (int i = 1; i< numero; i++) {
  if (numero % i == 0) {
    sumaDivisores += i;
  }
}

if (sumaDivisores == numero) {
  println("El número es perfecto");
} else {
  println("El número no es perfecto");
}
```

```
/* PROBLEMA 4.27
Un número abundante es un número x para el cual sig(x) >= 2x
donde sig(x) es la suma de todos los divisores de x (incluyendo el x)
Haz un programa que dado x, indique si es o no abundante
*/

int x = (int) random(0,101); println("x = "+x);
int sumaDivisores = 0;

for (int i = 1; i<=x; i++){
  if (x%i == 0) {sumaDivisores += i;}
}
if (sumaDivisores >= (2*x)){
  println("El número es abundante");
} else
```

```
/*PROBLEMA 4.32
Haz un programa que calcule el factorial de un número natural dado n
*/
int n = (int) random(22, 31);
print(n + "! = " );
long factorial = 1; // La clave de este problema es usar long
for (int i = n; i >= 1; i--) {factorial *= i;}
print(factorial);
```

```
/* PROBLEMA 4.16
Dadas dos fechas (día, mes, año) determinar que día es posterior.
*/

int a1 = (int)random(0, 2025.5); int a2 = (int)random(0, 2025.5);
int m1 = (int)random(0, 12.5); int m2 = (int)random(0, 12.5);
int d1 = (int)random(0,30.5); int d2 = (int)random(0, 30.5);

println("FECHA 1: " + d1 + "/" + m1 + "/" + a1);
println("FECHA 2: " + d2 + "/" + m2 + "/" + a2);

if (a1 < a2) {
                    println("FECHA 1 anterior a FECHA 2");
          } else if (a1 > a2) {
                     println("FECHA 1 posterior a FECHA 2");
          } else { // Los años son iguales

  if (m1 < m2) {
                     println("FECHA 1 anterior a FECHA 2");
          } else if (m1 > m2) {
                    println("FECHA 1 posterior a FECHA 2");
           } else { // Los años y meses son iguales

                     if (d1 < d2) {
                                println("FECHA 1 anterior a FECHA 2");
                        } else if (d1 > d2) {
                                println("FECHA 1 posterior a FECHA 2");
                         } else { // Los años, meses y días son iguales
                                println("FECHA 1 igual a FECHA 2");
                     }
           }
}
```

```
/* EJERCICIO DE CLASE (PARA USAR SWITCH)
Indica la calificación de un amigo en función de su nota.
*/
int nota = (int) random(0,11);println("Su nota es: " + nota);
switch (nota) {
    case 10: println("Sobresaliente alto"); break;
    case 9: println("Sobresaliente"); break;
    case 8: println("Notable alto"); break;
    case 7: println("Notable"); break;
    case 6: println("Bien"); break;
    case 5: println("Suficiente"); break;
    default: println("Insuficiente"); break;
}
```

```
/* PROBLEMA 4.30
(Mirar en apuntes, muy largo)
*/
String nombreVia = "";
final int VIA_URBANA_1_CARRIL = 0;
final int VIA_URBANA_2_CARRIL = 1;
final int CARRETERA_CONVECIONAL = 2;
final int AUTOVIA_AUTOPISTA = 3;
int tipoVia = (int) random(0,4);
final float BASE_MULTA = 200;

float velocidadMS = random(1, 40);
float velocidadKMH = velocidadMS * 3.6;
println("La velocidad (km/h) es: " + velocidadKMH);
float velocidadMaxima = 0;

switch (tipoVia) {
  case VIA_URBANA_1_CARRIL:
    println("El tipo de vía es: VÍA URBANA (1 CARRIL)");
    velocidadMaxima = 30;
    break;
  case VIA_URBANA_2_CARRIL:
    println("El tipo de vía es: VÍA URBANA (+1 CARRIL)");
    velocidadMaxima = 50;
    break;
  case CARRETERA_CONVECIONAL:
    println("El tipo de vía es: CARRETERA CONVENCIONAL");
    velocidadMaxima = 90;
    break;
  case AUTOVIA_AUTOPISTA:
    println("El tipo de vía es: AUTOPISTA");
    velocidadMaxima = 120;
    break;
}
print("Velocidad máxima permitida: " + velocidadMaxima);

float multa = 0;
if (velocidadKMH > velocidadMaxima) {
  println("HAY MULTA");
  multa = BASE_MULTA * (velocidadKMH / velocidadMaxima);
  println("Precio: " + multa+ "€");
} else {
  println("NO HAY MULTA");
}
```

TEMA 5: FUNCIONES

Una función es una herramienta que dados un número finito de parámetros, devuelve otro valor después de ejecutar una serie de líneas de código. Además podemos crear "procesos", que son funciones que no devuelven ningún valor. Lo hacemos con la palabra reservada *void*.

tipo_de_dato_que_devuelve nombre_función (param1, param2, param3, ...) {

Instrucciones a ejecutar con los parámetros

return valor_que_devuelve;

}

nombre_funcion(arg1, arg2, arg3) // A LOS PARÁMETROS DE LA FUNCIÓN LES PASAMOS ARGUMENTOS

```
/*EJEMPLO DE CLASE
FUNCIÓN QUE DADOS DOS NÚMEROS, LOS SUMA
*/
int sumarNumeros (int a; int b) {
          return a+b
}
int resultado = sumarNumeros(3,5)
```

Cuando trabajamos con funciones tenemos que crear una función *setup(); que "arranca el programa" (el código se ejecuta en el setup()*.

void setup () {

Código a ejecutar para arrancar el programa

}

```
/*PROBLEMA 5.10
Define una función que, usando la función random(0,1) devuelva un números
enteros a y b dados. Prueba a ejecutar tu función N veces para a = 30 y b = 50
*/
float numeroAleatorio (int min, int max) {
  return min + random(0,1) * (max-min);
}

void setup(){
  int N = 20;
  for (int i = 1; i < N; i++){
    println(numeroAleatorio(30,50));
  }
}
```

Podemos pedir datos al usuario por pantalla así como mostrar ventanas (ver apuntes del tema 5) pero para elllo tenemos que usar una librería externa (no entra en el exámen).

```
/*PROBLEMA 5.13
Define una función que muestra por pantalla los divisores de un número natural n dado. Prueba tu función N veces (N=15)
Para un valor de n aleatorio entre 2 y 200;
*/
String divisores(int n) {
  String listaDivisores = "";
  for (int i = 1; i<=n; i++) {
    if (n%i == 0) {
      listaDivisores = listaDivisores + i + ", ";
    }
  }
  return listaDivisores;
}

void setup() {
  int N  = 3;
  int numero;
  for (int j = 0; j < N; j++) {
    numero = (int)random(2,200);
    println("n = " + numero + "| " +  divisores(numero));
  }
}
```

```
/*PROBLEMA 5.16
Dada una cadena de texto determinar si es o no un palíndromo, es decir si se lee igual de derecha a izquierda que de izquierda
a derecha.
*/
String machacarTexto(String texto) {
  // Devuelve el texto sin espacios
  String cadenaComplementaria = "";
  for (int i = 0; i<texto.length(); i++) {
    if (texto.charAt(i) != ' ') {
      cadenaComplementaria += texto.charAt(i);
    }
  }
  return cadenaComplementaria;
}
boolean esPalindromo(String texto) {
  // Machacar el texto (quitarle los espacios)
  texto = machacarTexto(texto);

  int longitud = texto.length();
  int posicion = 0;
  boolean palindromo = true;

  while ( palindromo && (posicion <= ( longitud / 2 ) )) {
    if ( texto.charAt(posicion) != texto.charAt(longitud - posicion - 1) ) {
      palindromo = false;
    }
    posicion++;
  }
  if (palindromo) {
    return palindromo;
  }
  return false;
}

void setup() {
  println("reconocer: " + esPalindromo("reconocer"));
  println("hoy: " + esPalindromo("hoy"));
  println("yo hago yoga hoy: " + esPalindromo("yo hago yoga hoy"));

}
```

Hay algunas funciones especiales que nos permiten trabajar en Strings:

- Texto.charAt(posición) devuelve el caractér en la posición 'posición' del String. (¡Posiciones empiezan en cero!)
- Texto.length() devuelve la longitud del String

```
/*PROBLEMA 18

Haz una función que, dados tres puntos (xi,yi) calcule y devuelva el área del triángulo que conforma.

En caso de que esos tres puntos no conformen un triángulo según el Teorema de la desigualdad, la función devolverá

-1 como "código de error".

*/


float distancia(float x1, float y1, float x2, float y2) {

  return sqrt( sq(x1-x2) + sq(y1-y2) );

}


float calcularArea(float x1, float y1, float x2, float y2, float x3, float y3) {

  float lado1 = distancia(x1,y1,x2,y2);

  float lado2 = distancia(x1,y1,x3,y3);

  float lado3 = distancia(x2,y2,x3,y3);

  float semiperimetro = (lado1+lado2+lado3) / 2;


  if (lado1 + lado2 <= lado3) { // Por el teorema de la desigualdad, no es un triángulo.

    return -1;

  }

  return sqrt(semiperimetro * (semiperimetro-lado1)* (semiperimetro-lado2)* (semiperimetro-lado3));


}


void setup(){

  final float x1 = random(0,10);

  final float x2 = random(0,10);

  final float x3 = random(0,10);

  final float y1 = random(0,10);

  final float y2 = random(0,10);

  final float y3 = random(0,10);


  println("El área es: " + calcularArea(x1,y1,x2,y2,x3,y3));


}
```

```
/*EJERCICIO CLASE
Escribe un programa que dado un número natural, escriba una a una sus cifras.

La idea es que, n = 1234
n %10 == 4
(n/10) %10 == 3
(n/100) %10 == 2
y así sucesivamente
*/
int n = 12345;

while (n>0) {
  println("Cifra: " + n%10);
  n = n/10;
}
```

```
/* PROBLEMA 5.19
Escribe una función que dado un número natural n devuelva otro número que sea igual
a la suma de las cifras de n que ocupan una posición par menos la suma de las cifras que ocupan una posición impar
(Unidades --> 1, decenas --> 2)
*/

int n = 5013;
int suma = 0;
int cifra = 0;
int posicionCifra = 1;

while (n > 0) {
  cifra = n%10;
  n = n / 10; // Pasar a la siguiente cifra

  if (posicionCifra % 2 == 0) { // Cifra en posción par
    suma += cifra;
  } else { // Cifra en posicion impar
    suma -= cifra;
  }
  posicionCifra++;
}
println("Suma: " + suma);
```

```
/* PROBLEMA 5.20
 Dado n >= 3 entero, entonces no existen enteros positivos x^n + y^n = z^n

 Haz una función que dados dos números naturales n y max, compruebe si se cumple o no el
 Teorema de Fermat para cualesquiera z,y,z en [1, max]
 */

boolean seCumpleFermat(int n, int max) {
  boolean seCumple = true;
  for (int x = 1; x<= max; x++) {
    for (int y = 1; y<=max; y++) {
      for (int z = 1; z<=max; z++) {
        if ( pow(x, n) + pow(y, n) == pow(z, n)) {
          seCumple = false;
          println("Contraejemplo al Teorema de Fermat");
        }
      }
    }
  }
  return seCumple;
}

void setup() {
  // Probamos la función con n en {3,4} ý max en {25, 50, 100}
  for (int n = 3; n <= 4; n++) {
    for (int max = 25; max <= 100; max = max * 2) {
      seCumpleFermat(n, max);
    }
  }
}
```

```
/*PROBLEMA CLASE
Un número es curioso si:
- Es capicúa
- Es múltiplo de la suma de sus dígitos
Haz un programa que determine si un número es o no curioso
*/

boolean esCurioso (int numero) {return (esCapicua(numero) && (numero%sumaCifras(numero) == 0)); }

boolean esCapicua (int numero) {return reversoNumero(numero) == numero;}

// ALGORITMO IMPORTANTE PARA DARLE LA VUELTA A UN NÚMERO
int reversoNumero(int numero) {
  // 123 --> 321
  // Algoritmo: ((3*10)+2)*10 + 1
  int reverso = 0;
  int digito = 0;
  while (numero > 0) {
    digito = numero % 10;
    reverso = reverso*10 + digito;
    numero = numero / 10;
  }
  return reverso;
}

int sumaCifras(int numero) {
  int suma = 0;
  while(numero > 0) {
    suma+=numero%10;
    numero = numero / 10;
  }
  return suma;
}

void setup() {
  // Generar aleatorios hasta que encuentre N curiosos entre [min y max]
  int numeroDeCuriososEncontrados = 0; // Numero de curiosos que he encontrado
  int N = 5; // Número de curiosos que quiero encontrar
  int min = 6;
  int max = 200;
  while (numeroDeCuriososEncontrados  < N) {
    int numero = (int) random(min, max);
    if (esCurioso(numero)){
      numeroDeCuriososEncontrados++;
      println("El número " + numero + " es el número (" +numeroDeCuriososEncontrados+ ") curiso");
    }
  }
}
```

TEMA 6: DATOS ESTRUCTURADOS

A veces los datos simples no son suficientes. Por ejemplo, codificar un punto n-dimensional (n-upla). Trabajaremos con tipos de datos estáticos (la memoria que tienen reservada no cambia en el tiempo). Hay estructuras de datos dinámicas. En tiempo de ejecución pueden cambiar su memoria.

Aliasing → Las variables declaradas en una función y los argumentos de otra función (aunque coincidan en valores) ocupan posiciones DIFERENTES de memoria. Por ejemplo:

```
void f (int a) {
        a = 37;
}
void setup()
        int b = 7;
        f(b); // DESPÚES DE INVOCAR A LA FUNCIÓN,  b SIGUE VALIENDO SIETE
}
```

Arrays Un array es una colección indexada de elementos (los elementos son un tipo particular de dato.) En los arrays los índices se mueven desde cero hasta n-1. Por ejemplo, un Array de siete elementos:

0	1	2	3	4	5	6

Para declarar una variable de tipo Array tenemos que hacer una reserva explícita de memoria (no como con los datos simples).

```
TipoDeDato[] NombreDeLaVariable; // DEFINICIÓN
NombreDeLaVariable = new TipoDeDato[NúmeroDeElementos] // RESERVA DE MEMORIA
```

Y se puede poner todo en la misma línea. Por ejemplo:

```
int[] a = new int[100]; // Array "a" para guardar 100 números enteros
```

Además puedo tener funciones que devuelven un Array así como funciones que modifican Arrays (los reciben como argumento).

```
Int[] funcionQueDevuelveUnArray(argumentos) {
        Int [] arrayQueDevuelve = new int[ Tamaño];
        Instrucciones
        Return arrayQueDevuelve;
}


void funcionQueTrabajaConUnArray(arrayQueModifica, otros argumentos) {
        // Instrucciones que modifican el array
}
```

EN LA SEGUNDA FUNCIÓN SÍ ESTOY MODIFICANDO EL ARRAY EXTERNAMENTE (ALIASING).

```
/* EJEMPLO DE CLASE: Escribe una función que genera una lista aleatoria*/
int[] generarListaAleatoria(int numeroDeElementos){
  int[] lista = new int[numeroDeElementos];
  for (int i = 0; i < numeroDeElementos; i++) {
    lista[i] = (int) random(0,100);
  }
  return lista;
}
void setup() { int[] l = generarListaAleatoria(5); }
```

En realidad los **Strings** son también datos estructurados (array de caracteres)

char [] caracteres = {'h', 'o', 'l', 'a'}; es equivalente a *String caracteresStr = "hola";*

Consultar el array *caracteres[3]* es equivalente a caracteresStr.CharAt(3);

Los Strings son inmutables, es decir no puedo cambiar el carácter en posición [i]

```
void setup() {
  String cadena = "tu mamá es *inserte adjetivo*";
  for (int i = 0; i < cadena.length(); i++) {
    println(cadena.charAt(i));
  }
}
```

```
/* PROBLEMA DE CLASE
 Dado un String de '(' y ')' contar cuántos paréntesis bien creados hay
 */

String crearParentesisAleatorios(int longitud) {
  String cadena = "";
  for (int i = 0; i < longitud; i++) {
    if (random(0, 1) < 0.5) {
      cadena += ")";
    } else {
      cadena += "(";
    }
  }
  return cadena;
}

int parentesisCorrectos(String cadena) {
  int contadorDeParentesisCorrectos = 0;
  int contadorAbiertos = 0;
  for (int i = 0; i < cadena.length (); i++) {
    if (cadena.charAt(i) == '(') {
      contadorAbiertos++;
    } else if ((cadena.charAt(i) == ')') && (contadorAbiertos > 0)) {
      contadorAbiertos--;
      contadorDeParentesisCorrectos++;
    }
  }
  return contadorDeParentesisCorrectos;
}
```

El array se relacciona con los vectores en matemáticas, análogamente podemos pensar en matrices (arrays de arrays).

```
void setup() {
        int[] array1D = new int [10]; // 10-upla
        int[][] array2D = new int[10][5]; // Matriz 10x5
        int[][][][]...[] arrayND = new int [a][b][c][d]...[z] // Matriz axbxcxd...xz
}
```

Y para acceder a la posición (i,j) lo escribo como *array2D[i][j];*

```
void setup() {
  String cadena = "tu mamá es *inserte adjetivo*";
  for (int i = 0; i < cadena.length(); i++) {
    println(cadena.charAt(i));
  }
}
```

```
/*PROBLEMA CLASE
 Escribe una función que genere una matriz de un rango dado
 */


int[][] generarMatriz(int filas, int columnas, int min, int max) {
  int[][] matriz = new int [filas] [columnas];
  for (int i = 0; i < filas; i ++ ) {
    for (int j = 0; j < columnas; j++) {
      matriz[i][j] = (int) random(min, max);
    }
  }
  return matriz;
}


void imprimirBienMatriz(int[][] m) {
  for (int i = 0; i < m.length; i++) {
    for (int j = 0; j < m[i].length; j++) {
      print("  " + m[i][j] +"  ");
    }
  println("");
  }
}


void setup() {
  int[][] m = generarMatriz(10, 5, 3, 10);
  imprimirBienMatriz(m);
}
```

```
/* PROBLEMA DE CLASE
 Calcular la suma de dos matrices, si es posible
 */
// FUNCIONES AUXILIARES COMO LAS DEL EJERCICIO ANTERIOR
float[][] generarMatriz(int filas, int columnas, int min, int max) {
  float[][] matriz = new float [filas] [columnas];
  for (int i = 0; i < filas; i ++ ) {
    for (int j = 0; j < columnas; j++) {
      matriz[i][j] = random(min, max);
    }
  }
  return matriz;
}

void imprimirBienMatriz(float[][] m) {
  for (int i = 0; i < m.length; i++) {
    for (int j = 0; j < m[i].length; j++) {
      print("  " + m[i][j] +"  ");
    }
  println("");
  }
}
float[][] sumaMatrices (float[][]matriz1, float[][]matriz2) {
  if (matriz1 == null || matriz2 == null) {
    return null;
  }
  // Dos matrices se pueden sumar si tienen la misma dimensión
  else if (matriz1.length != matriz2.length || (matriz1[0].length != matriz2[0].length)) {
    println("Las dimensiones no son iguales");
    return null;
  }
  float[][] suma = new float[matriz1.length][matriz1[0].length];
  for (int i = 0; i < matriz1.length; i++) {
    for (int j = 0; j < matriz1[0].length; j++) {
      suma[i][j] = matriz1[i][j] + matriz2[i][j];
    }
  }
  return suma;
}
void setup() {
  float[][] m1 = generarMatriz(3,4,0,10);
  float[][] m2 = generarMatriz(3,4,0,10);
  println("MATRIZ M1: ");
  imprimirBienMatriz(m1);
  println("\nMATRIZ M2: ");
  imprimirBienMatriz(m2);
  println("\nMATRIZ M1+M2: ");
  imprimirBienMatriz(sumaMatrices(m1, m2));
  }
```

```
/* PROBLEMA DESAFÍO (PICADA CON MARCOS)
Dada una matriz de orden N, calcular su determinante
*/

int[][] quitarColumna(int[][] matriz, int pos) {
  int[][] resultado = new int [matriz.length][matriz[0].length-1];

  for (int i = 0; i < matriz.length; i++) {
    for (int j = 0; j < matriz[i].length; j++) {
      if (j < pos) {
        resultado[i][j] = matriz[i][j];
      } else if (j > pos) {
        resultado[i][j-1] = matriz[i][j];
      }
    }
  }
  return resultado;
}

int [][] quitarFila(int[][] matriz, int pos) {
  int[][] resultado = new int [matriz.length-1][matriz[0].length];
  for (int i = 0; i < matriz.length; i++) {
    if (i < pos) {
      resultado[i] = matriz[i];
    } else if (i > pos) {
      resultado[i-1] = matriz[i];
    }
  }
  return resultado;
}

int[][] menorMatricial (int[][] m, int fila, int columna) {
  return quitarColumna(quitarFila(m, fila), columna);
}

int det(int [][] m) {
   if (m.length == 1) {
    return m[0][0];
  }
  int resultado = 0;
  final int COL = 0; // Columna por la que desarrollo
  for (int fila = 0; fila < m.length; fila ++) {
    // DET = a00 * Adj00 + a10 * Adj10 + a20 * Adj20 + .... + aN0 * Adj N0
    resultado += m[fila][COL] * pow(-1, fila+COL) * det(menorMatricial(m, fila, COL));
  }
  return resultado;
}
```

Registros

Agrupar en un nuevo tipo de dato distintos campos de distintos tipos de datos.

```
class NombreDelRegistro {
        tipo de dato 1 nombreCampo1;
        tipo de dato 2 nombreCampo2;
        tipo de dato 3 nombreCampo3;
}
```

Por ejemplo:

```
class Persona {
        String nombre;
        String apellidos;
        int edad;
        char genero;
        String profesion;
        Int numeroDeKills;
}
Persona p1 = new Persona(); ; // NUEVO OBJETO CUYO TIPO DE DATO ES PERSONA
Persona p2 = new Persona();
Persona p3 = new Persona();
```

Para acceder a los campos tengo que usar un “.”:

```
p1.nombre = “Adolfo”;
p1.apellidos = “Jilgueros ”;
p1.edad = 42;
p1.genero = ‘H’;
p1.profesion = “político”;
p1.numeroDeKills = 65 000 000;
p2.nombre = “Jesús”;
p2.apellidos = “De Nazareth ”;
p2.edad = 33;
        p2.genero = ‘H’;
p2.profesion = “futuro carpintero, mesías”;
p2.numeroDeKills = -1;

p2.nombre = “Manolo”;
p2.apellidos = “Saorín Castaño”;
p2.edad =88;
```

p2.genero = “Creador”;

p2.profesion = “omnipotente”;

p2.numeroDeKills = $2^{\aleph_0}$*;*

Un ejemplo ciertamente más útil es:

```
class Fecha {
        int dia;
        int mes;
        int ano;
}
```

```
/*PROBLEMA DE CLASE
Crear un registro que permita manejar alumnos de clases (con sus nombres, apellidos, edad y nota)
*/

class Alumno {
  String nombre;
  String apellido;
  int edad;
  float nota;
}
String[] nombres = {"José", "Pepe", "Francisco", "David", "Marcos"};
String[] apellidos = {"Martínez", "López", "González", "Santos", "Carrillo", "Gil"};

Alumno crearAlumnoAleatorio() {
  Alumno alumno = new Alumno();
  alumno.nombre = nombres[(int) random(0, nombres.length)];
  alumno.apellido = apellidos[(int) random(0, apellidos.length)];
  alumno.edad = (int) random(18, 26);
  alumno.nota = random(0, 10);
  return alumno;
}

Alumno[] crearListaDeAlumnos(int numeroDeAlumnos) {
  Alumno[] listaDeAlumnos = new Alumno[numeroDeAlumnos];
  for (int i = 0; i < listaDeAlumnos.length; i++) {
    listaDeAlumnos[i] = crearAlumnoAleatorio();
  }
  return listaDeAlumnos;
}
void imprimirListaDeAlumnos(Alumno[] listaDeAlumnos) {
  for (int alumno = 0; alumno < listaDeAlumnos.length; alumno++) {
    Alumno a = listaDeAlumnos[alumno];
    println(a.nombre + " " + a.apellido + " (" + a.edad + ") nota: " + a.nota);
  }
}
```

TEMA 7: ALGORITMOS DE BÚSQUEDA Y ORDENACIÓN

Cuando las listas no están ordenadas empleamos búsqueda secuencial. Podemos también crear algoritmos para ordenar listas (que suelen ser mucho más interesantes). Los elementos de una lista pueden estar ordenados de forma ascendente o descendente. En general para clasificar cómo de bueno es un algoritmo utilizamos la nomenclatura $O(f(n))$ donde n es el número de elementos de la lista. Hay cuatro algoritmos principales de ordenación:

1. Burbuja → Estándar, tiempo cuadrático
    a. Intercambiar en parejas, una a una.
2. Selección → Estándar, el mejor para el caso general, tiempo cuadrático
    a. Buscar el elemento mínimo en la lista, intercambiarlo por el primero
    b. Buscar el segundo elemento mínimo (en lo que queda de lista), intercambiarlo por el segundo
3. Inserción → Estándar, muy bueno para cuando el caso es favorable, tiempo cuadrático
    a. Coje un elemento, Lo compara con todos los anteriores y lo pone en la posición en la que entra.
    b. Repite para el resto de elementos.
4. Ordenación por casilleros → Más eficiente en general.
    a. Crea casilleros o carpetas vacías
    b. Clasifica los elementos de la lista en estos casilleros
    c. Ordena cada uno de los casilleros
    d. Junta los casilleros para que todo esté ordenado al fina

```
/*PROBLEMA DE CLASE
 Dado una lista de alumnos como la programada en el tema de registros, ordenar la lista en función de sus notas.
 */
void ordenar(Alumno[] alumnos) {
  for (int i = 1; i < alumnos.length-1; i++) {
    for (int j = 0; j<alumnos.length-i; j++) {
      if (alumnos[j].nota > alumnos[i].nota) {
        Alumno alumnoAuxiliar = alumnos[j];
        alumnos[j] = alumnos[j+1];
        alumnos[j+1] = alumnoAuxiliar;
      }
    }
  }
}

void setup() {
  Alumno[] clase = crearListaDeAlumnos(5);
  imprimirListaDeAlumnos(clase);
  ordenar(clase);
  println("");
  imprimirListaDeAlumnos(clase);
}
```

```
/*TEORÍA (IMPLEMENTACIÓN DE LOS TRES ALGORITMOS) Ver mejor en el folio con todo explicado.*/
void burbuja(int[] lista) {
  for (int i = 0; i < lista.length-1; i++) {
    // FUNDAMENTAL LA DECLARACIÓN DE ESTE FOR
    for (int j = 0; j < lista.length-i-1; j++) {
      if (lista[j+1] > lista[j]) {
        int aux = lista[j];
        lista[j] = lista[j+1];
        lista[j+1] = aux;
      }
    }
  }
}

void seleccion (int[] lista) {
  for (int i = 0; i < lista.length-1; i++) {
    int posicionMaximo = i;
    for (int j = i+1; j<lista.length; j++) {
      // ERROR TÍPICO, comparar lista[j] con lista[i] en lugar de con lista[posicionMaximo], que es el que guarda el máximo
      if (lista[j] > lista[posicionMaximo]) {
        posicionMaximo = j;
      }
    }
    // INTERCAMBIO LA POSICION DE I POR LA POSICION DEL MÁXIMO
    int aux = lista[i];
    lista[i] = lista[posicionMaximo];
    lista[posicionMaximo] = aux;
  }
}

void insercion(int[] lista) {
  for (int i = 1; i < lista.length; i++) {
    int valorActual = lista[i];
    int posicion = i-1;

    while (posicion >= 0 && lista[posicion] >= valorActual) {
      // ERROR PONER AQUÍ ++posicion PORQUE LUEGO HAGO posicion-- Y ROMPERÍA TODO
      lista[posicion+1] = lista[posicion];
      posicion--;
    }
    lista[++posicion] = valorActual;
  }
}
```

TEMA 8: RECURSIVIDAD

Funciones que se llaman a sí mismas. Por ejemplo, el factorial viene definido en términos del factorial anterior:

$$n! = n(n-1)!;\ (n-1)! = (n-1)(n-2)!$$

Es importante que todas las funciones recursivas tengan un caso base para así en cierto momento, que escapen de la recursividad. Es fundamental que toda función recursiva tenga dos partes, recursión y caso base.

```
/*PROBLEMA DE CLASE
Calcular a el factorial de un numero
*/
int factorial(int n) {
  if (n <= 1) {
    return 1;
  }
  return n * factorial (n-1);
}
```

```
/*PROBLEMA DE CLASE
Calcular a el enésimo número de fibonacci
*/
int fib(int n) {
  if (n == 0 || n == 1) {
    return 1;
  }
  return fib(n-1) * fib(n-2);
}
```

```
/*PROBLEMA DE CLASE
Calcular la suma de los primeros n naturales.
*/
int suma(int n) {
  if (n == 1) {
    return 1;
  }
  return n + suma(n-1);
}
void setup () {
  println(suma(10));
}
```

```
/*PROBLEMA DE CLASE
Calcular la suma de los primeros n naturales.
*/
/*PROBLEMA DE CLASE
 Dada una lista de números enteros, contar el número de veces que aparece un número dado usando
 recursividad.
 */

int contar(int[] lista, int n) {
  if (lista.length == 1) {
    if (lista[0] == n) {
      return 1;
    } else {
      return 0;
    }
  }
  int[] lis = new int [lista.length -1];
  for (int i = 0; i < lis.length; i++) {
    lis[i] = lista[i+1];
  }

  if (lista[0] == n) {
    return 1 + contar(lis, n);
  } else {
    return contar(lis, n);
  }
}

void setup() {
  int[] l = {2, 2, 2, 4, 2, 1}; // 4
  println(contar(l, 2));
}
```

```
/*PROBLEMA DE CLASE
 Dado un número 'n' implementa un algoritmo recursivo que crea una pirámide de asteriscos.

 Por ejemplo, para n = 5:

 *
 * *
 * * *
 * * * *
 */

void asteriscos(int t) {
  String texto = "";
  for (int i = 0; i  < t; i ++) {
    texto+="*";
  }
  println(texto);
}
void imprimir(int n, int nivel) {
  if (nivel == n) {
    asteriscos(nivel);
  } else {
    asteriscos(nivel);
    imprimir(n, nivel+1);
  }
}

void setup() {
  imprimir(10, 1);
}
```

-

```
/*PROBLEMA DE PRÁCTICAS

 Dado un array de enteros, recorrerla de forma recursiva y sumar todos sus elementos.

 Se presentan cuatro formas recursivas de sumar.

 */


int sumaRecursivaCasoBaseFinal(int[] lista, int posicion, int suma) {

  // Comienza por la izquierda y el caso base es cuando se sale por la derecha.

  // --> ............................. FINAL

  if (posicion == lista.length) { // Es decir, me he salido

    return suma;

  }


  suma += lista[posicion]; // Incremento

  // TENGO QUE USAR EL ++posicion y no posicion++ PORQUE QUIERO QUE LO INCREMENTE A UNO Y LUEGO LO MANDE A LA FUNCIÓN (PRE-INCREMENTO FRENTE A POST-INCREMENTO)

  return sumaRecursivaCasoBaseFinal(lista, ++posicion, suma);

}


int sumaRecursivaCasoBaseInicial(int[] lista, int posicion, int suma) {

  // Comienza por la derecha y el caso base es cuando se sale por la izquierda

  if (posicion == -1) {

    return suma + lista[posicion];

  }

  suma = suma + lista[posicion];

  return sumaRecursivaCasoBaseInicial(lista, posicion-1, suma);

}



void setup() {

  int[] lista = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

  println(sumaRecursivaCasoBaseFinal(lista, 0, 0));

}
```

TEMA 9 y 10: INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A OBJETOS

Nuevo paradigma. En una clase agrupamos todos los elementos que definen un concepto. Las funciones dentro de una clase se llaman meétodos. Así las clases son composiciones de atributos y métodos.

Ejemplo:

- Clase: círculo
  - Atributos: radio, centro, color
  - Métodos: área, perímetro
- Objetos: diferentes círculos (cada uno con su radio y su perímetro)

```
/* PROBLEMA DE CLASE
 Crea una clase que permita almacenar datos según su radio y su color
 */

class Circulo {
  float radio;
  float centroX;
  float centroY;
  float centroZ;

  float area() {
    // No recibe nada porque todos los atributos de una clase son accesibles desde dentro de los métodos de dicha clase
    return PI * sq(radio);
  }

  float perimetro() {
    return PI * 2 * radio;
  }
}

void setup() {
  Circulo circulo1 = new Circulo();
  circulo1.radio = 2;
  circulo1.centroX = 0;
  circulo1.centroY = 0;
  circulo1.centroZ = 0;
  println("Área: " + circulo1.area());
  println("Perímetro: " + circulo1.perimetro());
}
```

```
/* PROBLEMA DE CLASE
 Crea una clase para representar un punto que permita calcular la distancia desde ese punto hasta otro
 */

class Punto {
  float x;
  float y;
  float z;

  float distanciaEuclidea(Punto punto) {
    return sqrt(sq(x-punto.x) + sq(y-punto.y) + sq(z-punto.z));
  }
}

void setup() {
  Punto p1 = new Punto();
  p1.x = 0;
  p1.y = 0;
  p1.z = 0;
  Punto p2 = new Punto();
  p2.x = 1;
  p2.y = 1;
  p2.z = 0;
  println(p1.distanciaEuclidea(p2));
}
```

El constructor es un método que se llama igual que la clase y que nos permite crear objetos sin necesidad de una función”crearAlgo” externa. Así podemos inicializar los objetos directamente. Se nombra con el mismo nombre que la clase:

```
/* PROBLEMA DE CLASE:
Modela utilizando POO una cuenta bancaria.
*/
 class CuentaBancaria {
  String titular;
  String numeroDeCuenta;
  float saldo;
  float limiteRetirada;

  // CONSTRUCTOR
  CuentaBancaria(String titular, String numeroDeCuenta, float limiteRetirada) {
    this.titular = titular;
    this.numeroDeCuenta = numeroDeCuenta;
    this.limiteRetirada = limiteRetirada;
    this.saldo = 0;
  }
  // MÉTODOS
  void ingresarDinero(float cantidad) {
    if (cantidad <= 0) {
      println("La cantidad ha de ser menor de cero");
    }
    saldo+=cantidad;
  }

  void retirarDinero(float cantidad) {
    if (cantidad < 0 || cantidad > limiteRetirada || cantidad > saldo) {
      println("Error en la retirada");
    }
    saldo-=cantidad;
  }

  void mostrarInformacion() {
    println(titular + "[" + numeroDeCuenta + "]" + "   (" + saldo+ "€)   Limite retirada: " + limiteRetirada );
  }
}

void setup() {
  CuentaBancaria cuenta = new CuentaBancaria("Sergio", "ES1325", 30);
  cuenta.mostrarInformacion();
  cuenta.ingresarDinero(20);
  cuenta.mostrarInformacion();
}
```