



GOBIERNO DE
MÉXICO

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO®

INSTITUTO TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CIUDAD MADERO

Tarea No. 4

Kotlin 2.0 Ya Tutorial

Pérez Anastasio Karla Zafiro - 20070574

Garay Hernandez Miguel Enrique - 20070600

Programación Nativa Para Móviles

09:00 - 10:00

Jorge Peralta Escobar

Marzo del 2025

Ciudad Madero, Tamaulipas, México.



ÍNDICE

Repository:	12
1. Tipos de variables.....	12
Ejercicio 1 - Hola Mundo.....	12
Ejecución.....	12
Ejercicio 2 - Dos valores.....	12
Ejecución.....	13
Ejercicio 2 - Conciso.....	13
Ejecución.....	14
Problemas propuestos.....	14
Ejercicio 3 - Variable inmutable.....	14
Ejecución.....	14
Ejercicio 4 - Tres variables inmutables.....	14
Ejecución.....	15
2. Entrada de datos por teclado en la Consola.....	15
Ejercicio 5 - Entrada de datos.....	15
Ejecución.....	16
Ejercicio 6 - Perímetro del cuadrado.....	16
Ejecución.....	16
Ejercicio 7 - Total a pagar de productos.....	16
Ejecución.....	17
Problemas propuestos.....	17
Ejercicio 8 - Suma y producto de números.....	17
Ejecución.....	19
Ejercicio 9 - Suma y promedio de números.....	19
Ejecución.....	20
3. Estructura condicional if.....	20
3.1 Estructura condicional simple.....	20
Ejercicio 10 - Impuestos.....	20
Ejecución.....	21
3.2 Estructura condicional compuesta.....	21
Ejercicio 11 - Valor mayor.....	21
Ejecución.....	21
3.3 Operadores.....	22
Ejercicio 12 - Producto y división de valores.....	22
Ejecución.....	23
Problemas propuestos.....	23
Ejercicio 13 - Promocionado.....	23
Ejecución.....	24
Ejercicio 14 - Uno o dos dígitos de un valor.....	24
Ejecución.....	24
4. Estructura condicional if como expresión.....	25
Ejercicio 15 - Mayor entre dos números.....	25
Ejecución.....	25

Ejercicio 16 - Cuadrado o cubo de número par o impar.....	25
Ejecución.....	26
Problema propuesto.....	26
Ejercicio 17 - Cantidad de dígitos de un valor.....	26
Ejecución.....	27
5. Estructuras condicionales anidadas.....	27
Ejercicio 18 - Promocionado, regular o libre.....	27
Ejecución.....	28
Problemas propuestos.....	28
Ejercicio 19 - Mayor de tres números distintos.....	29
Ejecución.....	30
Ejercicio 20 - Positivo, nulo o negativo.....	30
Ejecución.....	31
Ejercicio 21 - Uno, dos o tres cifras de un valor.....	31
Ejecución.....	32
Ejercicio 22 - Capacitación.....	32
Ejecución.....	33
6. Condiciones compuestas con operadores lógicos.....	33
Operador &&.....	33
Ejercicio 23 - Valor máximo de tres números.....	33
Ejecución.....	34
Operador 	34
Ejercicio 24 - Trimestres.....	34
Ejecución.....	35
Problemas propuestos.....	35
Ejercicio 25 - Navidad.....	35
Ejecución.....	36
Ejercicio 26 - Calcular el cubo de tres valores iguales.....	36
Ejecución.....	37
Ejercicio 27 - Todos los números menores a diez.....	37
Ejecución.....	38
Ejercicio 28 - Algún número menor a diez.....	38
Ejecución.....	38
Ejercicio 29 - Coordenadas y cuadrantes.....	39
Ejecución.....	39
Ejercicio 30 - Mayor y menor.....	39
Ejecución.....	40
7. Estructura repetitiva while.....	40
Ejercicio 31 - While.....	40
Ejecución.....	41
Ejercicio 32 - Serie de números.....	41
Ejecución.....	42
Ejercicio 33 - Promedio de diez valores.....	42
Ejecución.....	43

Ejercicio 34 - Piezas.....	43
Ejecución.....	44
Problemas propuestos.....	44
Ejercicio 35 - Notas de alumnos.....	44
Ejecución.....	45
Ejercicio 36 - Alturas.....	45
Ejecución.....	46
Ejercicio 37 - Sueldos de empleados.....	46
Ejecución.....	47
Ejercicio 38 - Veinticinco términos.....	47
Ejecución.....	48
Ejercicio 39 - Listas.....	48
Ejecución.....	48
Ejercicio 40 - Mayor de dos listas.....	49
Ejecución.....	50
Ejercicio 41 - Cuántos pares e impares hay.....	50
Ejecución.....	51
8. Estructura repetitiva do/while.....	51
Ejercicio 42 - Cuántos dígitos tiene el número.....	51
Ejecución.....	52
Ejercicio 43 - Promedio de valores hasta ingresar el cero.....	52
Ejecución.....	53
Ejercicio 44 - Piezas aptas.....	53
Ejecución.....	54
Problemas propuestos.....	54
Ejercicio 45 - Suma hasta teclear 9999.....	55
Ejecución.....	56
Ejercicio 46 - Acreedor, deudor o nulo.....	56
Ejecución.....	57
9. Estructura repetitiva for y expresiones de rango.....	57
Ejercicio 47 - Imprimir números hasta el cien.....	57
Ejecución.....	58
Ejercicio 48 - Imprimir números hasta el cien.....	58
Ejecución.....	59
Ejercicio 49 - Notas mayores o menores.....	59
Ejecución.....	60
Ejercicio 50 - Cantidad de múltiplos.....	60
Ejecución.....	62
Ejercicio 51 - Cantidad de pares.....	62
Ejecución.....	62
Problemas propuestos.....	63
Ejercicio 52 - Triángulos.....	63
Ejecución.....	64
Ejercicio 53 - Suma de los últimos cinco valores.....	64

Ejecución.....	65
Ejercicio 54 - Tabla del cinco.....	65
Ejecución.....	65
Ejercicio 55 - Tabla de multiplicar.....	65
Ejecución.....	66
Ejercicio 56 - Contador de Triángulos.....	66
Ejecución.....	67
Ejercicio 57 - Puntos en cuadrantes.....	68
Ejecución.....	69
Ejercicio 58 - Cantidad de negativos, positivos, múltiplos de 15 y suma de pares.....	69
Ejecución.....	70
10. Estructura condicional when.....	71
Ejercicio 59 - En qué eje se encuentra el punto.....	71
Ejecución.....	71
Ejercicio 60 - Promocionado, regular o libre.....	71
Ejecución.....	72
Ejercicio 61 - Piezas procesadas.....	72
Ejecución.....	73
Ejercicio 62 - Sueldo bajo, medio o alto.....	73
Ejecución.....	74
Problemas propuestos.....	75
Ejercicio 63 - Cero, positivo o negativo.....	75
Ejecución.....	76
Ejercicio 64 - Acumular el mayor de cada lista de 3 valores.....	76
Ejecución.....	77
Ejercicio 65 - Cantidad de triángulos.....	77
Ejecución.....	79
Ejercicio 66 - Número en texto.....	79
Ejecución.....	79
Ejercicio 67 - Cuántos dígitos tiene el número.....	80
Ejecución.....	80
Ejercicio 68 - Uno, cinco, diez o cero.....	80
Ejecución.....	81
Problema propuesto.....	81
Ejercicio 69 - Hijos en la familia.....	81
Ejecución.....	82
11. Concepto de funciones.....	83
Ejercicio 70 - Función suma de valores.....	83
Ejecución.....	84
Ejercicio 71 - Función suma de valores cinco veces.....	84
Ejecución.....	85
Problemas propuestos.....	85
Ejercicio 72 - Cuadrado y producto de valores.....	85
Ejecución.....	86

Ejercicio 73 - Función menor valor.....	86
Ejecución.....	87
12. Funciones: parámetros.....	88
Ejercicio 74 - Función que calcula la suma de los valores ingresados.....	88
Ejecución.....	89
Ejercicio 75 - Función mayor de tres valores.....	89
Ejecución.....	90
Ejercicio 76 - Función perímetro o superficie.....	90
Ejecución.....	91
Problemas propuestos.....	91
Ejercicio 77 - Función verificar claves.....	91
Ejecución.....	92
Ejercicio 78 - Función ordenar menor a mayor.....	92
Ejecución.....	93
13. Funciones: con retorno de datos.....	94
Ejercicio 79 - Función que calcula y retorna la superficie de un cuadrado.....	94
Ejercicio 80 - Función que retorna el mayor de dos valores.....	94
Ejecución.....	95
Ejercicio 81 - Función nombre más largo.....	95
Ejecución.....	96
Problemas propuestos.....	96
Ejercicio 82 - Función promedio de tres números.....	96
Ejecución.....	97
Ejercicio 83 - Función perímetro de un cuadrado.....	97
Ejecución.....	98
Ejercicio 84 - Función rectángulo de mayor superficie.....	98
Ejecución.....	99
14. Funciones: con retorno de datos.....	100
Ejercicio 85 - Función superficie de un cuadrado.....	100
Ejecución.....	100
Ejercicio 86 - Función retornarMayor.....	100
Ejecución.....	101
Ejercicio 87 - Función convertirCastelano.....	101
Ejecución.....	102
Problemas propuestos.....	102
Ejercicio 88 - Función retornarPromedio.....	102
Ejecución.....	103
Ejercicio 89 - Función retornarPerímetro.....	103
Ejecución.....	104
Ejercicio 90 - Función retornarSuperficie.....	104
Ejecución.....	105
Ejercicio 91 - Función largo.....	105
Ejecución.....	106
15. Funciones: con parámetros con valor por defecto.....	106

Ejercicio 92 - Función subrrayado.....	106
Ejecución.....	107
Problemas propuestos.....	107
Confeccionar una función que reciba entre 2 y 5 enteros. La misma nos debe retornar la suma de dichos valores. Debe tener tres parámetros por defecto.....	107
Ejercicio 93 - Función subrrayado.....	107
Ejecución.....	108
16. Funciones: llamada a la función con argumentos nombrados.....	108
Ejercicio 94 - Función sueldo.....	108
Ejecución.....	109
Problemas propuestos.....	109
Ejercicio 95 - Función tabla.....	109
Ejecución.....	110
17. Funciones: internas o locales.....	110
Ejercicio 96 - Función multiplo2y5.....	110
Ejecución.....	112
Problemas propuestos.....	112
Ejercicio 97 - Función Mayor.....	112
Ejecución.....	113
18. Funciones: conceptos.....	113
Ejercicio 98 - Sueldos función.....	113
Ejecución.....	114
Ejercicio 99 - Promedio de alturas.....	114
Ejecución.....	115
Ejercicio 100 - Arreglo menor a mayor.....	115
Ejecución.....	116
Ejercicio 101 - Arreglo primero y último.....	116
Ejecución.....	116
Ejercicio 102 - Diez elementos del arreglo.....	117
Ejecución.....	117
Problemas propuestos.....	117
Ejercicio 103 - Valor acumulado arreglo.....	118
Ejecución.....	119
Ejercicio 104 - Arreglo resultante.....	119
Ejecución.....	120
19. Funciones: parámetros y retorno de datos tipo arreglo.....	120
Ejercicio 105 - Elementos de un arreglo.....	120
Ejecución.....	121
Ejercicio 106 - Imprime sueldos de un arreglo.....	121
Ejecución.....	122
Problemas propuestos.....	122
Ejercicio 107 - Arreglar cargar.....	122
Ejecución.....	123
Ejercicio 108 - Arreglar cargar.....	124

Ejecución.....	125
20. POO -Conceptos de programación orientada a objetos.....	126
Ejercicio 109 - Clase persona.....	126
Ejecución.....	127
Ejercicio 110 - Clase triangulo.....	127
Ejecución.....	128
Problema propuesto.....	128
Ejercicio 111 - Clase Alumno.....	128
Ejecución.....	130
21. POO - Constructor de la clase.....	130
Ejercicio 112 - Clase Persona constructor.....	130
Ejecución.....	130
Ejercicio 113 - Clase Triangulo constructor.....	131
Ejecución.....	131
Ejercicio 114 - Clase Triangulo primario y secundario.....	131
Ejecución.....	132
Problemas propuestos.....	133
Ejercicio 115 - Clase Alumno y nota.....	133
Ejecución.....	134
Ejercicio 116 - Clase punto coordenada.....	134
Ejecución.....	135
22. POO - Llamada de métodos desde otro método de la misma clase.....	136
Ejercicio 117 - Clase Operaciones.....	136
Ejecución.....	136
Problema propuesto.....	136
Ejercicio 118 - Clase Hijos.....	137
Ejecución.....	138
23. POO - Colaboración de clases.....	138
Ejercicio 119 - Clase Cliente.....	138
Ejecución.....	139
Ejercicio 120 - Clase Dado.....	139
Ejecución.....	140
Problema propuesto.....	140
Ejercicio 121 - Clase socio.....	140
Ejecución.....	141
24. POO - modificadores de acceso private y public.....	141
Ejercicio 122 - Clase operaciones.....	141
Ejecución.....	142
Ejercicio 123 - Clase Dado impresion.....	142
Ejecución.....	142
Problema propuesto.....	143
Ejercicio 124 - Clase vector.....	143
Ejecución.....	144
25. POO - propiedades y sus métodos optionales set y get.....	144

Ejercicio 125 - Clase Persona personalizada.....	144
Problemas propuestos.....	145
Ejercicio 126 - Clase Empleados nombre y sueldo.....	145
Ejecución.....	146
Ejercicio 127 - Clase Dado valor incial.....	146
Ejecución.....	147
26. POO - data class.....	147
Ejercicio 128 - Data class articulo.....	147
Ejecución.....	148
Ejercicio 129 - Data class persona.....	148
Ejecución.....	149
Problema propuesto.....	149
Ejercicio 130 - Data class Dado valor.....	149
Ejecución.....	149
27. POO - enum class.....	149
Ejercicio 131 - Enum class TipoCarta.....	149
Ejecución.....	150
Ejercicio 132 - Enum class TipoOperacion.....	150
Ejecución.....	151
Problema propuesto.....	151
Ejercicio 133 - Enum class Paises.....	151
Ejecución.....	152
28. POO - objeto nombrado.....	152
Ejercicio 134 - Object Matematica.....	152
Ejecución.....	152
Ejercicio 135 - Objeto dados.....	152
Ejecución.....	153
Problema propuesto.....	153
Ejercicio 136 - Object Mayor.....	153
Ejecución.....	154
29. POO - herencia.....	154
Ejercicio 137 - Clase base Persona.....	154
Ejecución.....	155
Ejercicio 138 - open class Calculadora.....	155
Ejecución.....	156
Problema propuesto.....	156
Ejercicio 139 - open class Dado.....	156
Ejecución.....	157
30. POO - herencia - clases abstractas.....	157
Ejercicio 140 - open class Dado.....	157
Ejecución.....	158
Problema propuesto.....	158
Ejercicio 141 - abstract class Cuenta.....	159
Ejecución.....	160

31. POO - declaración e implementación de interfaces.....	160
Ejercicio 142 - interface Punto.....	160
Ejecución.....	161
Ejercicio 143 - interface Figura.....	161
Ejecución.....	162
Ejercicio 144 - Mayores de edad.....	162
Ejecución.....	162
Problemas propuestos.....	163
Ejercicio 145 - data class Articulo.....	163
Ejecución.....	164
Ejercicio 146 - class Dado mutable.....	164
Ejecución.....	165
32. Funciones de orden superior.....	165
Ejercicio 147 - funcion operar.....	165
Ejecución.....	166
Ejercicio 148 - class Persona nombre y edad.....	166
Ejecución.....	167
33. Expresiones lambda.....	167
Ejercicio 149 - class Persona mayoria de edad.....	167
Ejecución.....	168
Ejercicio 150 - funcion imprimirSi.....	168
Ejecución.....	169
Ejercicio 151 - función filtrar.....	169
Ejecución.....	170
34. Expresiones lambda con arreglos IntArray, FloatArray, DoubleArray etc.....	170
Ejercicio 152 - Elementos de un arreglo.....	170
Ejecución.....	170
Problema propuesto.....	171
Ejercicio 153 - función main arreglo de diez.....	171
Ejecución.....	172
35. Expresiones lambda con arreglos IntArray, FloatArray, DoubleArray etc.....	172
Ejercicio 154 - recorrer y procesar un arreglo.....	172
Ejecución.....	173
Ejercicio 155 - forEach.....	173
Ejercicio 156 - Arreglo de objetos Persona.....	173
Ejecución.....	174
Problema propuesto.....	174
Ejercicio 157 - Clase Dado valor y dos métodos.....	174
Ejecución.....	176
36. Funciones de extensión.....	176
Ejercicio 158 - Extensión de String.....	176
Ejecución.....	176
Ejercicio 159 - Extensión de IntArray.....	176
Ejecución.....	177

Problemas propuestos.....	177
Ejercicio 160 - función de extensión.....	177
Ejecución.....	177
Ejercicio 161 - funcion Int.hasta.....	177
Ejecución.....	178
37. Sobrecarga de operadores.....	178
Ejercicio 162 - clase Vector con operadores.....	178
Ejecución.....	180
Ejercicio 163 - clase Vector con sobrecarga.....	180
Ejecución.....	181
Ejercicio 164 - Arreglo de cinco enteros.....	181
Ejercicio 165 - Comparación de objetos.....	181
Ejecución.....	182
Ejercicio 166 - class TaTeTi.....	182
Ejecución.....	183
Ejercicio 167 - Clase Dados con invoke.....	183
Ejecución.....	184
Ejercicio 168 - Clase vector con +=.....	184
Ejecución.....	185
Ejercicio 169 - Operador in.....	185
Ejecución.....	185
38. Funciones: número variable de parámetros.....	185
Ejercicio 170 - Función número variable.....	185
Ejecución.....	186
Ejercicio 171 - enum y argumentos variables.....	186
Ejecución.....	186
Problema propuesto.....	186
Ejercicio 172 - función cantidadMayores.....	186
Ejecución.....	187
39. Colecciones - List y MutableList.....	187
Ejercicio 173 - Listas inmutables.....	187
Ejecución.....	188
Ejercicio 174 - Lista entrada por teclado.....	188
Ejecución.....	188
Ejercicio 175 - Operaciones con una lista inmutable.....	188
Ejecución.....	189
Ejercicio 176 - Lista de enteros aleatorios.....	190
Ejecución.....	190
Ejercicio 177 - Lista de objetos persona y conteo.....	190
Ejecución.....	191
Problemas propuestos.....	191
Ejercicio 178 - Lista de cien elementos aleatorios.....	191
Ejecución.....	192
Ejercicio 179 - clase Empleado nombre y sueldo.....	192

Ejecución.....	193
Ejercicio 180 - Función cargar.....	193
Ejecución.....	194
40. Colecciones - Map y MutableMap.....	194
Ejercicio 181 - Map en kotlin.....	194
Ejecución.....	195
Ejercicio 182 - Funciones para trabajar con Map.....	195
Ejecución.....	196
Ejercicio 183 - Diccionario castellano-inglés.....	196
Ejecución.....	197
Ejercicio 184 - Mapa de productos.....	197
Ejecución.....	198
Ejercicio 185 - Diccionario de alumnos.....	198
Ejecución.....	199
Problema propuesto.....	200
Ejercicio 186 - Clase de datos para fecha.....	200
Ejecución.....	201
41. Colecciones - Set y MutableSet.....	201
Ejercicio 187 - Operaciones con conjuntos mutables.....	201
Ejecución.....	202
Ejercicio 188 - Verificación de fechas.....	202
Ejecución.....	203
Problema propuesto.....	203
Ejercicio 189 - MutableSet almacena seis valores.....	203
Ejecución.....	204
42. Package e Import.....	205
Ejercicio 190 - Archivo con funciones.....	205
Ejecución.....	205
Problema propuesto.....	205
Ejercicio 191 - Cartón de lotería.....	205
Ejecución.....	207
43. Corrutinas.....	207
Ejercicio 192 - Corrutinas y retardos.....	207
Problema propuesto.....	208
Ejercicio 193 - Corrutinas y retardos 2.....	208
44. Bibliografía.....	209

Repositorio:

<https://github.com/Zafirows/Programacion-nativa>

1. Tipos de variables

Ejercicio 1 - Hola Mundo

```
fun main (parametro: Array<String>) {  
    // Esta es la función principal de Kotlin, el punto de entrada del  
    programa  
  
    // 'parametro' es un arreglo de cadenas que puede recibir argumentos desde  
    la línea de comandos  
  
    print("Hola Mundo")  
  
    // Imprime en pantalla el texto "Hola Mundo" sin salto de línea  
}
```

Ejecución

Hola Mundo

Ejercicio 2 - Dos valores

```
fun main(parametro: Array<String>) {  
    // Declaración de dos variables enteras sin inicializar (se inicializan  
    después)  
  
    val valor1: Int  
    val valor2: Int  
  
    // Asignación de valores a las variables  
    valor1 = 100  
    valor2 = 400  
  
    // Declaración de una variable para guardar resultados  
    var resultado: Int  
  
    // Se calcula la suma de valor1 y valor2  
    resultado = valor1 + valor2  
  
    // Se imprime el resultado de la suma  
    println("La suma de $valor1 + $valor2 es $resultado")
```

```

// Se calcula el producto de valor1 y valor2
resultado = valor1 * valor2
// Se imprime el resultado del producto
println("El producto de $valor1 * $valor2 es $resultado")
}

```

Ejecución

```

La suma de 100 + 400 es 500
El producto de 100 * 400 es 40000

```

Ejercicio 2 - Conciso

```

fun main(parametro: Array<String>) {
    // Se declara una constante entera llamada 'valor1' y se le asigna el
    valor 100
    val valor1: Int = 100

    // Se declara una constante entera llamada 'valor2' y se le asigna el
    valor 400
    val valor2: Int = 400

    // Se declara una variable llamada 'resultado' y se inicializa con la suma
    de valor1 y valor2
    var resultado: Int = valor1 + valor2

    // Se imprime el resultado de la suma usando interpolación de cadenas
    println("La suma de $valor1 + $valor2 es $resultado")

    // Ahora se actualiza la variable 'resultado' con el producto de valor1 y
    valor2
    resultado = valor1 * valor2

    // Se imprime el resultado de la multiplicación
    println("El producto de $valor1 * $valor2 es $resultado")
}

```

Ejecución

```
La suma de 100 + 400 es 500
El producto de 100 * 400 es 40000
```

Problemas propuestos

Definir una variable inmutable con el valor 50 que representa el lado de un cuadrado, en otras dos variables inmutables almacenar la superficie y el perímetro del cuadrado.

Mostrar la superficie y el perímetro por la Consola.

Definir tres variables inmutables y cargar por asignación los pesos de tres personas con valores Float. Calcular el promedio de pesos de las personas y mostrarlo.

Ejercicio 3 - Variable inmutable

```
fun main(argumento: Array<String>) {
    // Se declara una constante 'lado' que representa la longitud de un lado
    // del cuadrado
    val lado = 50

    // Se calcula el perímetro del cuadrado (4 veces el lado)
    val perimetro = lado * 4

    // Se calcula la superficie (área) del cuadrado (lado al cuadrado)
    val superficie = lado * lado

    // Se imprime el perímetro y la superficie del cuadrado en una sola línea
    print("El perímetro de un cuadrado de lado $lado es $perimetro y su
superficie es $superficie")
}
```

Ejecución

```
El perímetro de un cuadrado de lado 50 es 200 y su superficie es 2500
```

Ejercicio 4 - Tres variables inmutables

```
fun main(argumento: Array<String>) {
    // Se declaran tres variables de tipo Float con los pesos de tres personas
    val peso1 = 89.4f
```

```

val peso2 = 67f
val peso3 = 87.45f

// Se calcula el promedio de los tres pesos
val promedio = (peso1 + peso2 + peso3) / 3

// Se imprime el promedio de los pesos usando interpolación de cadenas
println("El promedio de los tres pesos de personas es $promedio")
}

```

Ejecución

El promedio de los tres pesos de personas es 81.28333

2. Entrada de datos por teclado en la Consola

Ejercicio 5 - Entrada de datos

```

fun main(argumento: Array<String>) {
    // Solicita al usuario que ingrese el primer valor
    print("Ingrese primer valor:")

    // Lee la entrada del usuario, la convierte a entero y la guarda en
    'valor1'

    val valor1 = readln().toInt()

    // Solicita al usuario que ingrese el segundo valor
    print("Ingrese segundo valor:")

    // Lee la entrada del usuario, la convierte a entero y la guarda en
    'valor2'

    val valor2 = readln().toInt()

    // Se calcula la suma de los dos valores ingresados
    val suma = valor1 + valor2

    // Se imprime el resultado de la suma
    println("La suma de $valor1 y $valor2 es $suma")
}

```

```

    // Se calcula el producto de los dos valores
    val producto = valor1 * valor2
    // Se imprime el resultado del producto
    println("El producto de $valor1 y $valor2 es $producto")
}

```

Ejecución

```

Ingresé primer valor:6
Ingresé segundo valor:7
La suma de 6 y 7 es 13
El producto de 6 y 7 es 42

```

Ejercicio 6 - Perímetro del cuadrado

```

fun main(parametro: Array<String>) {
    // Solicita al usuario que ingrese la medida de un lado del cuadrado
    print("Ingresé la medida del lado del cuadrado:")

    // Lee la entrada desde el teclado, la convierte a entero y la guarda en
    'lado'
    val lado = readln().toInt()

    // Calcula el perímetro del cuadrado multiplicando el lado por 4
    val perímetro = lado * 4

    // Muestra el resultado del perímetro en pantalla
    println("El perímetro del cuadrado es $perímetro")
}

```

Ejecución

```

Ingresé la medida del lado del cuadrado:20
El perímetro del cuadrado es 80

```

Ejercicio 7 - Total a pagar de productos

```

fun main(parametro: Array<String>) {
    // Solicita al usuario que ingrese el precio de un producto
}

```

```

print("Ingrese el precio del producto:")

    // Lee la entrada desde el teclado, la convierte a doble (decimal) y la
guarda en 'precio'

val precio = readln().toDouble()

    // Sigue al usuario que ingrese la cantidad de productos que desea
comprar

print("Ingrese la cantidad de productos:")

    // Lee la entrada desde el teclado, la convierte a entero y la guarda en
'cantidad'

val cantidad = readln().toInt()

    // Calcula el total a pagar multiplicando el precio por la cantidad de
productos

val total = precio * cantidad

    // Muestra el total a pagar en pantalla

println("El total a pagar es $total")
}

```

Ejecución

```

Ingrese el precio del producto:300
Ingrese la cantidad de productos:5
El total a pagar es 1500.0

```

Problemas propuestos

Escribir un programa en el cual se ingresen cuatro números enteros, calcular e informar la suma de los dos primeros y el producto del tercero y el cuarto.

Realizar un programa que lea por teclado cuatro valores numéricos enteros e informar su suma y promedio.

Ejercicio 8 - Suma y producto de números

```

fun main(parametro: Array<String>) {
    // Sigue al usuario que ingrese el primer valor

    print("Ingrese primer valor:")
}

```

```

// Lee la entrada, la convierte a entero y la guarda en 'v1'
val v1 = readln().toInt()

// Solicita al usuario que ingrese el segundo valor
print("Ingrese segundo valor:")

// Lee la entrada, la convierte a entero y la guarda en 'v2'
val v2 = readln().toInt()

// Solicita al usuario que ingrese el tercer valor
print("Ingrese tercer valor:")

// Lee la entrada, la convierte a entero y la guarda en 'v3'
val v3 = readln().toInt()

// Solicita al usuario que ingrese el cuarto valor
print("Ingrese cuarto valor:")

// Lee la entrada, la convierte a entero y la guarda en 'v4'
val v4 = readln().toInt()

// Se calcula la suma de los dos primeros valores
val suma = v1 + v2

// Se imprime el resultado de la suma
println("La suma de $v1 y $v2 es $suma")

// Se calcula el producto de los dos últimos valores
val producto = v3 * v4

// Se imprime el resultado del producto
println("El producto de $v3 y $v4 es $producto")
}

```

Ejecución

```
Ingrese primer valor:20
Ingrese segundo valor:30
Ingrese tercer valor:80
Ingrese cuarto valor:10
La suma de 20 y 30 es 50
El producto de 80 y 10 es 800
```

Ejercicio 9 - Suma y promedio de números

```
fun main(parametro: Array<String>) {
    // Solicita al usuario que ingrese el primer valor
    print("Ingrese primer valor:")

    // Lee la entrada, la convierte a entero y la guarda en 'v1'
    val v1 = readln().toInt()

    // Solicita al usuario que ingrese el segundo valor
    print("Ingrese segundo valor:")

    // Lee la entrada, la convierte a entero y la guarda en 'v2'
    val v2 = readln().toInt()

    // Solicita al usuario que ingrese el tercer valor
    print("Ingrese tercer valor:")

    // Lee la entrada, la convierte a entero y la guarda en 'v3'
    val v3 = readln().toInt()

    // Solicita al usuario que ingrese el cuarto valor
    print("Ingrese cuarto valor:")

    // Lee la entrada, la convierte a entero y la guarda en 'v4'
    val v4 = readln().toInt()

    // Se calcula la suma de los cuatro valores ingresados
    val suma = v1 + v2 + v3 + v4

    // Se imprime el resultado de la suma
    println("La suma de los cuatro valores es $suma")
```

```

    // Se calcula el promedio dividiendo la suma entre 4
    val prom = suma / 4
    // Se imprime el resultado del promedio
    println("El promedio de los cuatro valores es $prom")
}

```

Ejecución

```

Ingrese primer valor:10
Ingrese segundo valor:15
Ingrese tercer valor:6
Ingrese cuarto valor:2
La suma de los cuatro valores es 33
El promedio de los cuatro valores es 8

```

3. Estructura condicional if

3.1 Estructura condicional simple

Ejercicio 10 - Impuestos

```

fun main(parametro: Array<String>) {
    // Solicita al usuario que ingrese el sueldo del empleado
    print("Ingrese el sueldo del empleado:")

    // Lee la entrada desde el teclado y la convierte a tipo Double (para
    permitir decimales)

    val sueldo = readln().toDouble()

    // Verifica si el sueldo es mayor a 3000
    if (sueldo > 3000) {
        // Si el sueldo es mayor a 3000, imprime que debe pagar impuestos
        println("Debe pagar impuestos")
    }
}

```

Ejecución

```
Ingrese el sueldo del empleado:5000  
Debe pagar impuestos
```

3.2 Estructura condicional compuesta

Ejercicio 11 - Valor mayor

```
fun main(parametro: Array<String>) {  
  
    // Solicita al usuario que ingrese el primer valor  
    print("Ingrese primer valor:")  
  
    // Lee la entrada, la convierte a entero y la guarda en 'valor1'  
    val valor1 = readln().toInt()  
  
  
    // Solicita al usuario que ingrese el segundo valor  
    print("Ingrese segundo valor:")  
  
    // Lee la entrada, la convierte a entero y la guarda en 'valor2'  
    val valor2 = readln().toInt()  
  
  
    // Compara si 'valor1' es mayor que 'valor2'  
    if (valor1 > valor2)  
        // Si 'valor1' es mayor, imprime que el mayor valor es 'valor1'  
        print("El mayor valor es $valor1")  
    else  
        // Si no, imprime que el mayor valor es 'valor2'  
        print("El mayor valor es $valor2")  
}
```

Ejecución

```
Ingrese primer valor:33  
Ingrese segundo valor:35  
El mayor valor es 35
```

3.3 Operadores

Ejercicio 12 - Producto y división de valores

```
fun main(parametro: Array<String>) {  
    // Solicita al usuario que ingrese el primer valor  
    print("Ingrese el primer valor:")  
    // Lee la entrada, la convierte a entero y la guarda en 'valor1'  
    val valor1 = readln().toInt()  
  
    // Solicita al usuario que ingrese el segundo valor  
    print("Ingrese el segundo valor:")  
    // Lee la entrada, la convierte a entero y la guarda en 'valor2'  
    val valor2 = readln().toInt()  
  
    // Compara si 'valor1' es menor que 'valor2'  
    if (valor1 < valor2) {  
        // Si es verdadero, calcula la suma y la resta  
        val suma = valor1 + valor2  
        val resta = valor1 - valor2  
        // Imprime la suma y la resta de los dos valores  
        println("La suma de los dos valores es: $suma")  
        println("La resta de los dos valores es: $resta")  
    } else {  
        // Si no es menor, calcula el producto y la división  
        val producto = valor1 * valor2  
        val division = valor1 / valor2 // Este cálculo puede causar una  
        // excepción si 'valor2' es 0  
        // Imprime el producto y la división de los dos valores  
        println("El producto de los dos valores es: $producto")  
        println("La división de los dos valores es: $division")  
    }  
}
```

Ejecución

```
Ingrese el primer valor:23
Ingrese el segundo valor:14
El producto de los dos valores es: 322
La división de los dos valores es: 1
```

Problemas propuestos

Se ingresan tres notas de un alumno, si el promedio es mayor o igual a siete mostrar un mensaje "Promocionado".

Se ingresa por teclado un número entero comprendido entre 1 y 99, mostrar un mensaje indicando si el número tiene uno o dos dígitos.

(Tener en cuenta que condición debe cumplirse para tener dos dígitos, un número entero)

Ejercicio 13 - Promocionado

```
fun main(parametro: Array<String>) {
    // Solicita al usuario que ingrese la primera nota
    print("Ingrese primer nota:")
    // Lee la entrada, la convierte a entero y la guarda en 'nota1'
    val nota1 = readln().toInt()

    // Solicita al usuario que ingrese la segunda nota
    print("Ingrese segunda nota:")
    // Lee la entrada, la convierte a entero y la guarda en 'nota2'
    val nota2 = readln().toInt()

    // Solicita al usuario que ingrese la tercera nota
    print("Ingrese tercer nota:")
    // Lee la entrada, la convierte a entero y la guarda en 'nota3'
    val nota3 = readln().toInt()

    // Calcula el promedio de las tres notas
    var promedio = (nota1 + nota2 + nota3) / 3

    // Verifica si el promedio es mayor o igual a 7
```

```
if (promedio >= 7)
    // Si el promedio es mayor o igual a 7, imprime "Promocionado"
    println("Promocionado")
}
```

Ejecución

```
Ingrese primer nota:8
Ingrese segunda nota:9
Ingrese tercer nota:7
Promocionado
```

Ejercicio 14 - Uno o dos dígitos de un valor

```
fun main(parametro: Array<String>) {
    // Solicita al usuario que ingrese un valor comprendido entre 1 y 99
    print("Ingrese un valor comprendido entre 1 y 99:")

    // Lee la entrada y la convierte a entero, almacenándola en la variable
    'num'

    val num = readln().toInt()

    // Verifica si el número tiene menos de 10
    if (num < 10)
        // Si 'num' es menor que 10, significa que tiene un solo dígito
        println("Tiene un dígito")
    else
        // Si 'num' tiene 10 o más, significa que tiene dos dígitos
        println("Tiene dos dígitos")
}
```

Ejecución

```
Ingrese un valor comprendido entre 1 y 99:66
Tiene dos dígitos
```

4. Estructura condicional if como expresión

Ejercicio 15 - Mayor entre dos números

```
fun main(parametro: Array<String>) {  
    // Solicita al usuario que ingrese el primer valor  
    print("Ingrese primer valor:")  
    // Lee la entrada, la convierte a entero y la guarda en 'valor1'  
    val valor1 = readln().toInt()  
  
    // Solicita al usuario que ingrese el segundo valor  
    print("Ingrese segundo valor:")  
    // Lee la entrada, la convierte a entero y la guarda en 'valor2'  
    val valor2 = readln().toInt()  
  
    // Utiliza una expresión 'if' para determinar cuál de los dos valores es  
    mayor  
    val mayor = if (valor1 > valor2) valor1 else valor2  
  
    // Imprime el valor mayor entre 'valor1' y 'valor2'  
    println("El mayor entre $valor1 y $valor2 es $mayor")  
}
```

Ejecución

```
Ingrese primer valor:15  
Ingrese segundo valor:12  
El mayor entre 15 y 12 es 15
```

Ejercicio 16 - Cuadrado o cubo de número par o impar

```
fun main(parametro: Array<String>) {  
    // Solicita al usuario que ingrese un valor entero  
    print("Ingrese un valor entero: ")  
    // Lee la entrada, la convierte a entero y la guarda en 'valor'  
    val valor = readln().toInt()
```

```

// Se evalúa si el número ingresado es par o impar
val resultado = if (valor % 2 == 0) {
    // Si el valor es par, se calcula su cuadrado
    println("Cuadrado:")
    valor * valor
} else {
    // Si el valor es impar, se calcula su cubo
    println("Cubo:")
    valor * valor * valor
}

// Imprime el resultado, que será el cuadrado o cubo dependiendo del caso
println(resultado)
}

```

Ejecución

```

Ingrese un valor entero: 40
Cuadrado:
1600

```

Problema propuesto

Cargar un valor entero por teclado comprendido entre 1 y 99. Almacenar en otra variable la cantidad de dígitos que tiene el valor ingresado por teclado.

Mostrar la cantidad de dígitos del número ingresado por teclado.

Ejercicio 17 - Cantidad de dígitos de un valor

```

fun main(parametro: Array<String>) {
    // Solicita al usuario que ingrese un valor entero entre 1 y 99
    print("Ingrese un valor entero comprendido entre 1 y 99:")
    // Lee la entrada del usuario, la convierte a entero y la guarda en
    'valor'
    val valor = readln().toInt()

    // Se utiliza una expresión 'if' para determinar la cantidad de dígitos
}

```

```

    // Si el valor es menor que 10, tiene 1 dígito, de lo contrario tiene 2
    // dígitos

    var cantidad = if (valor < 10) 1 else 2

    // Imprime la cantidad de dígitos del número ingresado
    println("El número $valor tiene $cantidad dígito/s")
}

```

Ejecución

```

Ingrese un valor entero comprendido entre 1 y 99:5
El número 5 tiene 1 dígito/s

```

5. Estructuras condicionales anidadas

Ejercicio 18 - Promocionado, regular o libre

```

fun main(parametros: Array<String>) {
    // Solicita al usuario que ingrese la primera nota
    print("Ingrese primer nota:")

    // Lee la entrada, la convierte a entero y la guarda en 'nota1'
    val nota1 = readln().toInt()

    // Solicita al usuario que ingrese la segunda nota
    print("Ingrese segunda nota:")

    // Lee la entrada, la convierte a entero y la guarda en 'nota2'
    val nota2 = readln().toInt()

    // Solicita al usuario que ingrese la tercera nota
    print("Ingrese tercer nota:")

    // Lee la entrada, la convierte a entero y la guarda en 'nota3'
    val nota3 = readln().toInt()

    // Calcula el promedio de las tres notas
    val promedio = (nota1 + nota2 + nota3) / 3
}

```

```

// Se evalúa el promedio para determinar el estado del alumno
if (promedio >= 7)
    // Si el promedio es mayor o igual a 7, se imprime "Promocionado"
    print("Promocionado")
else
    // Si el promedio es menor a 7 pero mayor o igual a 4, se imprime
    "Regular"
    if (promedio >= 4)
        print("Regular")
    else
        // Si el promedio es menor a 4, se imprime "Libre"
        print("Libre")
}

```

Ejecución

```

Ingresé primer nota:9
Ingresé segunda nota:6
Ingresé tercer nota:4
Regular

```

Problemas propuestos

Se cargan por teclado tres números distintos. Mostrar por pantalla el mayor de ellos.

Se ingresa por teclado un valor entero, mostrar una leyenda que indique si el número es positivo, nulo o negativo.

Confeccionar un programa que permita cargar un número entero positivo de hasta tres cifras y muestre un mensaje indicando si tiene 1, 2, o 3 cifras. Mostrar un mensaje de error si el número de cifras es mayor.

Un postulante a un empleo, realiza un test de capacitación, se obtuvo la siguiente información: cantidad total de preguntas que se le realizaron y la cantidad de preguntas que contestó correctamente. Se pide confeccionar un programa que ingrese los dos datos por teclado e informe el nivel del mismo según el porcentaje de respuestas correctas que ha obtenido, y sabiendo que:

Nivel máximo: Porcentaje $\geq 90\%$.

Nivel medio: Porcentaje $\geq 75\% \text{ y } < 90\%$.

Nivel regular: Porcentaje $\geq 50\% \text{ y } < 75\%$.

Fuera de nivel: Porcentaje<50%.

Ejercicio 19 - Mayor de tres números distintos

```
fun main(parametro: Array<String>) {  
    // Solicita al usuario que ingrese el primer valor  
    print("Ingrese primer valor:")  
    // Lee el valor ingresado y lo convierte a entero, guardándolo en 'valor1'  
    val valor1 = readln().toInt()  
  
    // Solicita al usuario que ingrese el segundo valor  
    print("Ingrese segundo valor:")  
    // Lee el valor ingresado y lo convierte a entero, guardándolo en 'valor2'  
    val valor2 = readln().toInt()  
  
    // Solicita al usuario que ingrese el tercer valor  
    print("Ingrese tercer valor:")  
    // Lee el valor ingresado y lo convierte a entero, guardándolo en 'valor3'  
    val valor3 = readln().toInt()  
  
    // Compara los tres valores para encontrar el mayor  
    if (valor1 > valor2) {  
        // Si 'valor1' es mayor que 'valor2', entonces se compara con 'valor3'  
        if (valor1 > valor3)  
            // Si 'valor1' es mayor que 'valor3', entonces es el mayor valor  
            print(valor1)  
        else  
            // Si 'valor3' es mayor que 'valor1', entonces 'valor3' es el mayor  
            print(valor3)  
    } else {  
        // Si 'valor2' es mayor que 'valor1', entonces se compara con 'valor3'  
        if (valor2 > valor3)  
            // Si 'valor2' es mayor que 'valor3', entonces es el mayor valor  
            print(valor2)  
        else  
            // Si 'valor3' es mayor que 'valor2', entonces 'valor3' es el mayor  
            print(valor3)  
    }  
}
```

```
    }  
}  
}
```

Ejecución

```
Ingrese primer valor:30  
Ingrese segundo valor:60  
Ingrese tercer valor:14  
60
```

Ejercicio 20 - Positivo, nulo o negativo

```
fun main(parametro: Array<String>) {  
    // Solicita al usuario que ingrese un valor entero  
    print("Ingrese un valor entero:")  
    // Lee el valor ingresado y lo convierte a entero, guardándolo en 'valor'  
    val valor = readln().toInt()  
  
    // Compara el valor ingresado  
    if (valor == 0)  
        // Si el valor es igual a 0, imprime "Se ingresó el cero"  
        println("Se ingresó el cero")  
    else  
        // Si el valor no es 0, entra en una nueva comparación  
        if (valor > 0)  
            // Si el valor es mayor que 0, imprime "Se ingresó un valor positivo"  
            println("Se ingresó un valor positivo")  
        else  
            // Si el valor es menor que 0, imprime "Se ingresó un valor negativo"  
            println("Se ingresó un valor negativo")  
}
```

Ejecución

```
Ingrese un valor entero:30  
Se ingresó un valor positivo
```

```
Ingrese un valor entero:-15
Se ingresó un valor negativo
```

Ejercicio 21 - Uno, dos o tres cifras de un valor

```
fun main(parametro: Array<String>) {
    // Solicita al usuario que ingrese un valor entero con 1, 2 o 3 cifras
    print("Ingrese un valor entero con 1,2 o 3 cifras:")
    // Lee el valor ingresado y lo convierte a entero, guardándolo en 'valor'
    val valor = readln().toInt()

    // Compara el valor ingresado para determinar la cantidad de dígitos
    if (valor < 10)
        // Si el valor tiene solo un dígito (menor que 10), imprime "Tiene un
        dígito"
        println("Tiene un dígito")
    else
        // Si el valor tiene dos dígitos (menor que 100), imprime "Tiene dos
        dígitos"
        if (valor < 100)
            println("Tiene dos dígitos")
        else
            // Si el valor tiene tres dígitos (menor que 1000), imprime "Tiene
            tres dígitos"
            if (valor < 1000)
                println("Tiene tres dígitos")
            else
                // Si el valor es mayor o igual a 1000, imprime un error en la
                entrada de datos
                println("Error en la entrada de datos.")
}
```

Ejecución

```
Ingrese un valor entero con 1,2 o 3 cifras:666
Tiene tres dígitos
```

```
Ingrese un valor entero con 1,2 o 3 cifras:21
Tiene dos dígitos
```

Ejercicio 22 - Capacitación

```
fun main(parametro: Array<String>) {
    // Solicita al usuario ingresar la cantidad total de preguntas del examen
    print("Ingrese la cantidad total de preguntas del examen:")
    // Lee la cantidad total de preguntas y la guarda en la variable
    'totalPreguntas'
    val totalPreguntas = readln().toInt()

    // Solicita al usuario ingresar la cantidad de preguntas respondidas
    // correctamente
    print("Ingrese la cantidad total de preguntas contestadas correctamente:")
    // Lee la cantidad de preguntas correctas y la guarda en la variable
    'totalCorrectas'
    val totalCorrectas = readln().toInt()

    // Calcula el porcentaje de respuestas correctas
    val porcentaje = totalCorrectas * 100 / totalPreguntas

    // Compara el porcentaje con diferentes umbrales para determinar el nivel
    if (porcentaje >= 90)
        // Si el porcentaje es mayor o igual a 90, el nivel es máximo
        println("Nivel máximo")
    else
        if (porcentaje >= 75)
            // Si el porcentaje es mayor o igual a 75, el nivel es medio
            System.out.print("Nivel medio")
        else
            if (porcentaje >= 50)
                // Si el porcentaje es mayor o igual a 50, el nivel es regular
                println("Nivel regular")
            else
                // Si el porcentaje es menor que 50, el nivel es fuera de nivel
                println("Fuera de nivel")
}
```

```
}
```

Ejecución

```
Ingrese la cantidad total de preguntas del examen:15
Ingrese la cantidad total de preguntas contestadas correctamente:11
Nivel regular
```

```
Ingrese la cantidad total de preguntas del examen:15
Ingrese la cantidad total de preguntas contestadas correctamente:14
Nivel máximo
```

6. Condiciones compuestas con operadores lógicos

Operador &&

Ejercicio 23 - Valor máximo de tres números

```
fun main(parametro: Array<String>) {
    // Solicitar al usuario ingresar tres valores
    print("Ingrese primer valor: ")
    val num1 = readln().toInt()
    print("Ingrese segundo valor: ")
    val num2 = readln().toInt()
    print("Ingrese tercer valor: ")
    val num3 = readln().toInt()

    // Encontrar el valor máximo entre los tres
    val mayor = when {
        num1 > num2 && num1 > num3 -> num1
        num2 > num3 -> num2
        else -> num3
    }

    // Imprimir el número mayor
```

```
    println("El número mayor es: $mayor")
}
```

Ejecución

```
Ingrese primer valor: 45
Ingrese segundo valor: 13
Ingrese tercer valor: 69
El número mayor es: 69
```

Operador ||

Ejercicio 24 - Trimestres

```
fun main(parametro: Array<String>) {
    // Solicitar al usuario ingresar el día, mes y año
    print("Ingrese día: ")
    val dia = readln().toInt()
    print("Ingrese mes: ")
    val mes = readln().toInt()
    print("Ingrese año: ")
    val año = readln().toInt()

    // Verificar si la fecha es válida
    if (mes in 1..12 && dia in 1..31) {
        // Verificar el trimestre correspondiente
        when (mes) {
            1, 2, 3 -> println("Corresponde al primer trimestre.")
            4, 5, 6 -> println("Corresponde al segundo trimestre.")
            7, 8, 9 -> println("Corresponde al tercer trimestre.")
            10, 11, 12 -> println("Corresponde al cuarto trimestre.")
            else -> println("Mes no válido.")
        }
    } else {
        println("Fecha no válida.")
    }
}
```

Ejecución

```
Ingrese día: 29
Ingrese mes: 8
Ingrese año: 2024
Corresponde al tercer trimestre.
```

Problemas propuestos

Realizar un programa que pida cargar una fecha cualquiera, luego verificar si dicha fecha corresponde a Navidad.

Se ingresan tres valores por teclado, si todos son iguales calcular el cubo del número y mostrarlo.

Se ingresan por teclado tres números, si todos los valores ingresados son menores a 10, imprimir en pantalla la leyenda "Todos los números son menores a diez".

Se ingresan por teclado tres números, si al menos uno de los valores ingresados es menor a 10, imprimir en pantalla la leyenda "Alguno de los números es menor a diez".

Escribir un programa que pida ingresar la coordenada de un punto en el plano, es decir dos valores enteros x e y (distintos a cero).Posteriormente imprimir en pantalla en que cuadrante se ubica dicho punto. (1º Cuadrante si $x > 0$ Y $y > 0$, 2º Cuadrante: $x < 0$ Y $y > 0$, etc.)

Escribir un programa en el cual: dada una lista de tres valores enteros ingresados por teclado se guarde en otras dos variables el menor y el mayor de esa lista. Utilizar el if como expresión para obtener el mayor y el menor.

Imprimir luego las dos variables.

Ejercicio 25 - Navidad

```
fun main(parametro: Array<String>) {
    print("Ingrese día: ")
    val dia = readln().toInt() // Se solicita al usuario ingresar el día, y se convierte a tipo entero.

    print("Ingrese mes: ")
    val mes = readln().toInt() // Se solicita al usuario ingresar el mes, y se convierte a tipo entero.

    print("Ingrese año: ")
    val año = readln().toInt() // Se solicita al usuario ingresar el año. Este dato no se usa, pero se pide para completar la entrada.
```

```

    // Condicional que verifica si la fecha ingresada corresponde al 25 de
diciembre

    if (mes == 12 && dia == 25) {

        println("La fecha ingresada corresponde a Navidad.") // Si la fecha
es 25 de diciembre, imprime este mensaje.

    } else {

        println("La fecha ingresada no corresponde a Navidad.") // Si no es
25 de diciembre, imprime este mensaje.

    }

}

```

Ejecución

```

Ingrese día: 25
Ingrese mes: 10
Ingrese año: 2025
La fecha ingresada no corresponde a Navidad.

```

```

Ingrese día: 25
Ingrese mes: 12
Ingrese año: 2024
La fecha ingresada corresponde a Navidad.

```

Ejercicio 26 - Calcular el cubo de tres valores iguales

```

fun main(parametro: Array<String>) {

    print("Ingrese primer valor:")

    val valor1 = readln().toInt() // Leemos el primer valor e intentamos
convertirlo a un número entero

    print("Ingrese segundo valor:")

    val valor2 = readln().toInt() // Leemos el segundo valor e intentamos
convertirlo a un número entero

    print("Ingrese tercer valor:")

    val valor3 = readln().toInt() // Leemos el tercer valor e intentamos
convertirlo a un número entero

    // Condicional para verificar si los tres valores son iguales

    if (valor1 == valor2 && valor1 == valor3) {

        // Si los tres valores son iguales, se calcula el cubo del primer
valor (en este caso el cubo de valor1)
    }
}

```

```

        val cubo = valor1 * valor1 * valor3
        print("El cubo de $valor1 es $cubo") // Imprime el cubo calculado
    }
}

```

Ejecución

```

Ingrese primer valor:2
Ingrese segundo valor:2
Ingrese tercer valor:2
El cubo de 2 es 8

```

Ejercicio 27 - Todos los números menores a diez

```

fun main(parametro: Array<String>) {
    print("Ingrese primer valor:")

    val valor1 = readln().toInt() // El primer valor ingresado por el
    usuario se almacena en la variable valor1.

    print("Ingrese segundo valor:")

    val valor2 = readln().toInt() // El segundo valor ingresado por el
    usuario se almacena en la variable valor2.

    print("Ingrese tercer valor:")

    val valor3 = readln().toInt() // El tercer valor ingresado por el
    usuario se almacena en la variable valor3.

    // Condición que verifica si los tres valores son menores a 10
    if (valor1 < 10 && valor2 < 10 && valor3 < 10)

        println("Todos los números son menores a diez") // Si la condición
    es verdadera, imprime este mensaje.
}

```

Ejecución

```

Ingrese primer valor:3
Ingrese segundo valor:5
Ingrese tercer valor:6
Todos los números son menores a diez

```

Ejercicio 28 - Algún número menor a diez

```
fun main(parametro: Array<String>) {  
    print("Ingrese primer valor:") // Muestra un mensaje pidiendo el primer  
    valor.  
  
    val valor1 = readln().toInt() // Lee el primer valor ingresado por el  
    usuario y lo convierte a entero.  
  
    print("Ingrese segundo valor:") // Muestra un mensaje pidiendo el segundo  
    valor.  
  
    val valor2 = readln().toInt() // Lee el segundo valor ingresado por el  
    usuario y lo convierte a entero.  
  
    print("Ingrese tercer valor:") // Muestra un mensaje pidiendo el tercer  
    valor.  
  
    val valor3 = readln().toInt() // Lee el tercer valor ingresado por el  
    usuario y lo convierte a entero.  
  
    // La condición if verifica si alguno de los tres valores es menor que 10.  
    if (valor1 < 10 || valor2 < 10 || valor3 < 10)  
        print("Alguno de los números es menor a diez") // Si la condición es  
    verdadera, imprime este mensaje.  
}
```

Ejecución

```
Ingrese primer valor:30  
Ingrese segundo valor:12  
Ingrese tercer valor:1  
Alguno de los números es menor a diez
```

Ejercicio 29 - Coordenadas y cuadrantes

```
fun main(parametro: Array<String>) {  
    // Solicitar al usuario las coordenadas x y y  
    print("Ingrese coordenada x:")  
  
    val x = readln().toInt() // Leer la coordenada x y convertirla a entero  
  
    print("Ingrese coordenada y:")  
  
    val y = readln().toInt() // Leer la coordenada y y convertirla a entero
```

```

// Verificar en qué cuadrante se encuentra el punto (x, y)

if (x > 0 && y > 0) // Primer cuadrante: x y ambos positivos
    print("Se encuentra en el primer cuadrante")

else
    if (x < 0 && y > 0) // Segundo cuadrante: x negativo, y positivo
        print("Se encuentra en el segundo cuadrante")

    else
        if (x < 0 && y < 0) // Tercer cuadrante: x y ambos negativos
            print("Se encuentra en el tercer cuadrante")

        else
            if (x > 0 && y < 0) // Cuarto cuadrante: x positivo, y
negativo
                print("Se encuentra en el cuarto cuadrante")

            else // Si no está en ningún cuadrante, debe estar en un eje
(x = 0 o y = 0)
                print("Se encuentra en un eje")
}

```

Ejecución

```

Ingrese coordenada x:90
Ingrese coordenada y:60
Se encuentra en el primer cuadrante

```

Ejercicio 30 - Mayor y menor

```

fun main(parametro: Array<String>) {
    // Se piden tres valores al usuario
    print("Ingrese primer valor:")
    val valor1 = readln().toInt() // Primer valor ingresado
    print("Ingrese segundo valor:")
    val valor2 = readln().toInt() // Segundo valor ingresado
    print("Ingrese tercer valor:")
    val valor3 = readln().toInt() // Tercer valor ingresado

    // Determinar el menor valor usando una estructura condicional if
    val menor = if (valor1 < valor2 && valor1 < valor3) valor1 else if (valor2
< valor3) valor2 else valor3
}

```

```

    // Determinar el mayor valor usando una estructura condicional if
    val mayor = if (valor1 > valor2 && valor1 > valor3) valor1 else if (valor2
> valor3) valor2 else valor3

    // Mostrar el resultado
    print("El mayor de la lista es $mayor y el menor $menor")
}

```

Ejecución

```

Ingresé primer valor:2
Ingresé segundo valor:6
Ingresé tercer valor:3
El mayor de la lista es 6 y el menor 2

```

7. Estructura repetitiva while

Ejercicio 31 - While

```

fun main(parametro: Array<String>) {
    var x = 1 // Se inicializa la variable 'x' con el valor 1
    while (x <= 100) { // El ciclo while se ejecutará mientras 'x' sea menor
o igual a 100
        println(x) // Imprime el valor actual de 'x'
        x = x + 1 // Incrementa 'x' en 1 después de cada iteración
    }
}

```

Ejecución

1	27	53	79
2	28	54	80
3	29	55	81
4	30	56	82
5	31	57	83
6	32	58	84
7	33	59	85
8	34	60	86
9	35	61	87
10	36	62	88
11	37	63	89
12	38	64	90
13	39	65	91

Ejercicio 32 - Serie de números

```
fun main(parametro: Array<String>) {
    print("Ingrese un valor:") // Pide al usuario que ingrese un valor
    val valor = readln().toInt() // Lee el valor ingresado y lo convierte a
un número entero (Int)
    var x = 1 // Inicializa la variable 'x' en 1

    while (x <= valor) { // El ciclo while continuará mientras 'x' sea menor
o igual a 'valor'
        println(x) // Imprime el valor actual de 'x'
        x = x + 1 // Incrementa 'x' en 1
    }
}
```

Ejecución

```
Ingrese un valor:5
```

```
1  
2  
3  
4  
5
```

Ejercicio 33 - Promedio de diez valores

```
fun main(parametro: Array<String>) {  
  
    var x = 1 // Contador inicializado en 1  
    var suma = 0 // Acumulador para la suma de los valores  
  
    while (x <= 10) { // Se repite 10 veces  
        print("Ingrese un valor:")  
        val valor = readln().toInt() // Lee un valor ingresado por el  
        usuario y lo convierte a entero  
        suma = suma + valor // Se acumula el valor ingresado  
        x = x + 1 // Se incrementa el contador  
    }  
  
    println("La suma de los 10 valores ingresados es $suma")  
  
    val promedio = suma / 10 // Calcula el promedio (división  
    entera)  
    println("El promedio es $promedio")  
}
```

Ejecución

```
Ingrese un valor:6
Ingrese un valor:9
Ingrese un valor:2
Ingrese un valor:11
Ingrese un valor:12
Ingrese un valor:1
Ingrese un valor:1
Ingrese un valor:0
Ingrese un valor:12
Ingrese un valor:20
La suma de los 10 valores ingresados es 74
El promedio es 7
```

Ejercicio 34 - Piezas

```
fun main(parametro: Array<String>) {
    print("Cuantas piezas procesará:")
    val n = readln().toInt() // El usuario indica cuántas piezas se
    van a procesar

    var x = 1 // Contador
    var cantidad = 0 // Contador de piezas aptas

    while (x <= n) { // Bucle que se repite n veces
        print("Ingrese la medida de la pieza:")
        val largo = readln().toDouble() // El usuario ingresa la medida de
        la pieza (decimal)

        if (largo >= 1.20 && largo <= 1.30) // Si la medida está en el rango
        aceptable
            cantidad = cantidad + 1 // Se incrementa el contador de
            piezas aptas

        x = x + 1 // Incrementa el contador de iteraciones
    }

    print("La cantidad de piezas aptas son: $cantidad")
}
```

Ejecución

```
Cuantas piezas procesará:3
Ingrese la medida de la pieza:1.21
Ingrese la medida de la pieza:1.26
Ingrese la medida de la pieza:1.66
La cantidad de piezas aptas son: 2
```

Problemas propuestos

Escribir un programa que solicite ingresar 10 notas de alumnos y nos informe cuántos tienen notas mayores o iguales a 7 y cuántos menores.

Se ingresan un conjunto de n alturas de personas por teclado (n se ingresa por teclado). Mostrar la altura promedio de las personas.

En una empresa trabajan n empleados cuyos sueldos oscilan entre \$100 y \$500, realizar un programa que lea los sueldos que cobra cada empleado e informe cuántos empleados cobran entre \$100 y \$300 y cuántos cobran más de \$300. Además el programa deberá informar el importe que gasta la empresa en sueldos al personal.

Realizar un programa que imprima 25 términos de la serie 11 - 22 - 33 - 44, etc. (No se ingresan valores por teclado)

Mostrar los múltiplos de 8 hasta el valor 500. Debe aparecer en pantalla 8 - 16 - 24, etc.

Realizar un programa que permita cargar dos listas de 5 valores cada una. Informar con un mensaje cual de las dos listas tiene un valor acumulado mayor (mensajes "Lista 1 mayor", "Lista 2 mayor", "Listas iguales")

Tener en cuenta que puede haber dos o más estructuras repetitivas en un algoritmo.

Desarrollar un programa que permita cargar n números enteros y luego nos informe cuántos valores fueron pares y cuántos impares.

Emplear el operador "%" en la condición de la estructura condicional:

```
if (valor % 2 == 0)      //Si el if se verifica verdadero luego es par.
```

Ejercicio 35 - Notas de alumnos

```
fun main(parametro: Array<String>) {
    var x = 1                      // Contador de alumnos
    var conta1 = 0                  // Contador de aprobados (nota >= 7)
    var conta2 = 0                  // Contador de reprobados (nota < 7)

    while (x <= 10) {              // Se repite 10 veces, una por cada alumno
```

```

print("Ingrese nota:")

val nota = readln().toInt() // Se ingresa una nota entera

if (nota >= 7)
    conta1 = conta1 + 1 // Suma a los aprobados
else
    conta2 = conta2 + 1 // Suma a los reprobados

x = x + 1 // Pasa al siguiente alumno
}

println("Cantidad de alumnos con notas mayores o iguales a 7: $conta1")
println("Cantidad de alumnos con notas menores a 7: $conta2")
}

```

Ejecución

```

Ingrese nota:6
Ingrese nota:5
Ingrese nota:6
Ingrese nota:5
Ingrese nota:9
Ingrese nota:9
Ingrese nota:7
Ingrese nota:6
Ingrese nota:6
Ingrese nota:7
Cantidad de alumnos con notas mayores o iguales a 7: 4
Cantidad de alumnos con notas menores a 7: 6

```

Ejercicio 36 - Alturas

```

fun main(parametro: Array<String>) {
    print("Cuantas alturas ingresará?:")
    val n = readln().toInt() // Cantidad de personas

    var x = 1 // Contador
    var suma = 0.0 // Acumulador para la suma de alturas

    while (x <= n) // Se repite n veces

```

```

print("Ingrese la altura de la persona (Ej: 1.76) :")
val altura = readln().toDouble() // Leer altura como número decimal
suma = suma + altura // Acumular la altura
x = x + 1 // Avanzar al siguiente ingreso
}

val promedio = suma / n // Calcular el promedio
println("Altura promedio: $promedio") // Mostrar resultado
}

```

Ejecución

```

Ingrese la altura de la persona (Ej: 1.76) :1.50
Ingrese la altura de la persona (Ej: 1.76) :1.60
Ingrese la altura de la persona (Ej: 1.76) :1.80
Ingrese la altura de la persona (Ej: 1.76) :1.65
Altura promedio: 1.6375000000000002

```

Ejercicio 37 - Sueldos de empleados

```

fun main(parametro: Array<String>) {
    print("Cuantos empleados tiene la empresa:")
    val n = readln().toInt() // Cantidad de empleados a registrar

    var x = 1 // Contador para el bucle
    var conta1 = 0 // Contador para sueldos entre 100 y 300
    var conta2 = 0 // Contador para sueldos mayores a 300
    var gastos = 0.0 // Acumulador de sueldos totales

    while (x <= n) { // Bucle que se repite n veces
        print("Ingrese el sueldo del empleado:")
        val sueldo = readln().toDouble() // Leer sueldo como número decimal

        if (sueldo <= 300)
            conta1 += 1 // Contar sueldo bajo o medio
        else
            conta2 += 1 // Contar sueldo alto
    }
}

```

```

        gastos += sueldo          // Acumular sueldo en el total
        x += 1
    }

    // Mostrar resultados
    println("Cantidad de empleados con sueldos entre 100 y 300: $conta1")
    println("Cantidad de empleados con sueldos mayor a 300: $conta2")
    System.out.print("Gastos total de la empresa en sueldos: $gastos")
}

```

Ejecución

```

Cuantos empleados tiene la empresa:10
Ingrese el sueldo del empleado:100
Ingrese el sueldo del empleado:200
Ingrese el sueldo del empleado:200
Ingrese el sueldo del empleado:300
Ingrese el sueldo del empleado:1000
Ingrese el sueldo del empleado:800
Ingrese el sueldo del empleado:300
Ingrese el sueldo del empleado:250
Ingrese el sueldo del empleado:290
Ingrese el sueldo del empleado:190
Cantidad de empleados con sueldos entre 100 y 300: 8
Cantidad de empleados con sueldos mayor a 300: 2
Gastos total de la empresa en sueldos: 3630.0

```

Ejercicio 38 - Veinticinco términos

```

fun main(parametro: Array<String>) {
    var termino = 11      // Primer número de la serie
    var x = 1             // Contador

    while (x <= 25) {    // Repite 25 veces
        println(termino)           // Imprime el número actual
        termino = termino + 11     // Suma 11 al número anterior
        x = x + 1                 // Incrementa el contador
    }
}

```

Ejecución

```
11  
22  
33  
44  
55  
66  
77  
88  
99  
110  
121  
132  
143  
154  
165  
176  
187  
198  
209  
220  
231  
242  
253  
264  
275
```

Ejercicio 39 - Listas

```
fun main(parametro: Array<String>) {  
    var mult8 = 8 // Comienza desde 8  
    while (mult8 <= 500) { // Mientras no pase de 500  
        print("$mult8 -") // Imprime el múltiplo seguido de un  
guion  
        mult8 = mult8 + 8 // Suma 8 para el siguiente múltiplo  
    }  
}
```

Ejecución

```
8 -16 -24 -32 -40 -48 -56 -64 -72 -80 -88 -96 -104 -112 -120 -128 -136  
-144 -152 -160 -168 -176 -184 -192 -200 -208 -216 -224 -232 -240 -248  
-256 -264 -272 -280 -288 -296 -304 -312 -320 -328 -336 -344 -352 -360  
-368 -376 -384 -392 -400 -408 -416 -424 -432 -440 -448 -456 -464 -472 -480 -488 -496
```

Ejercicio 40 - Mayor de dos listas

```
fun main(parametro: Array<String>) {  
    var x = 1                                // Contador para el primer bucle  
    var sumal = 0                             // Acumulador de la primera lista  
  
    println("Ingreso de la primer lista de valores") // Mensaje para el usuario  
  
    // Ingreso de los primeros 5 valores  
    while (x <= 5) {  
        print("Ingrese valor:")           // Sigue al usuario  
        val valor = readln().toInt()      // Lee y convierte el valor a entero  
        sumal = sumal + valor          // Acumula el valor en sumal  
        x = x + 1                      // Incrementa el contador  
    }  
  
    println("Ingreso de la segunda lista de valores") // Mensaje para segunda lista  
    x = 1                                // Reinicia el contador  
    var suma2 = 0                            // Acumulador de la segunda lista  
  
    // Ingreso de los siguientes 5 valores  
    while (x <= 5) {  
        print("Ingrese valor:")           // Sigue al usuario  
        val valor = readln().toInt()      // Lee y convierte a entero  
        suma2 = suma2 + valor          // Acumula en suma2  
        x = x + 1                      // Incrementa el contador  
    }  
  
    // Comparación de las sumas  
    if (sumal > suma2)  
        print("Lista 1 mayor.")        // Si sumal es mayor  
    else  
        if (suma2 > sumal)  
            print("Lista2 mayor.")     // Si suma2 es mayor  
        else
```

```

        print("Listas iguales.")    // Si ambas sumas son iguales
}

```

Ejecución

```

Ingreso de la primer lista de valores
Ingrese valor:50
Ingrese valor:30
Ingrese valor:12
Ingrese valor:54
Ingrese valor:60
Ingreso de la segunda lista de valores
Ingrese valor:13
Ingrese valor:15
Ingrese valor:30
Ingrese valor:44
Ingrese valor:32
Lista 1 mayor.

```

Ejercicio 41 - Cuántos pares e impares hay

```

fun main(parametro: Array<String>) {
    var pares = 0                      // Contador de números pares
    var impares = 0                     // Contador de números impares

    print("Cuantos números ingresará:")
    val n = readln().toInt()           // El usuario indica cuántos números va
a ingresar

    var x = 1                          // Contador de iteraciones (empezamos en
1)

    // Bucle que se repite hasta que se ingresen n números
    while (x <= n) {
        print("Ingrese el valor:")
        val valor = readln().toInt() // El usuario ingresa un número

        if (valor % 2 == 0)          // Verifica si el número es par
            pares = pares + 1       // Si es par, incrementa el contador de
pares
        else
    }
}

```

```

        impares = impares + 1      // Si no, incrementa el contador de
impares

        x = x + 1                  // Avanza al siguiente número
    }

// Muestra los resultados
println("Cantidad de pares: $pares")
println("Cantidad de impares: $impares")
}

```

Ejecución

```

Cuantos números ingresará:5
Ingrrese el valor:8
Ingrrese el valor:7
Ingrrese el valor:5
Ingrrese el valor:9
Ingrrese el valor:10
Cantidad de pares: 2
Cantidad de impares: 3

```

8. Estructura repetitiva do/while

Ejercicio 42 - Cuántos dígitos tiene el número

```

fun main(parametro: Array<String>) {
    do {
        // Sigue al usuario ingresar un valor comprendido entre 0 y 999
        print("Ingrrese un valor comprendido entre 0 y 999:")
        val valor = readln().toInt()      // Convierte el valor ingresado a un
número entero

        // Verifica cuántos dígitos tiene el número ingresado
        if (valor < 10)

            println("El valor ingresado tiene un dígito") // Si el número es
menor a 10, tiene un dígito
    }
}

```

```

        else
            if (valor < 100)
                println("El valor ingresado tiene dos dígitos") // Si es
menor a 100, tiene dos dígitos
            else
                println("El valor ingresado tiene tres dígitos") // Si es
menor a 1000, tiene tres dígitos

        } while (valor != 0) // El ciclo se repite hasta que el usuario ingrese
el valor 0
}

```

Ejecución

```

Ingrese un valor comprendido entre 0 y 999:321
El valor ingresado tiene tres dígitos
Ingrese un valor comprendido entre 0 y 999:666
El valor ingresado tiene tres dígitos
Ingrese un valor comprendido entre 0 y 999:50
El valor ingresado tiene dos dígitos
Ingrese un valor comprendido entre 0 y 999:3
El valor ingresado tiene un dígito
Ingrese un valor comprendido entre 0 y 999:0
El valor ingresado tiene un dígito

Process finished with exit code 0
|

```

Ejercicio 43 - Promedio de valores hasta ingresar el cero

```

fun main(parametro: Array<String>) {
    var cant = 0 // Contador de cuántos valores válidos ( $\neq 0$ ) se
ingresaron
    var suma = 0 // Acumulador para sumar los valores ingresados

    do {
        print("Ingrese un valor (0 para finalizar):")
        val valor = readln().toInt() // Leer valor desde consola y
convertirlo a entero

        // Si el valor ingresado no es 0, lo sumamos y aumentamos el contador
        if (valor != 0) {
            suma += valor // suma = suma + valor
            cant++ // cant = cant + 1
        }
    }
}

```

```

        }

    } while (valor != 0)          // Repetimos mientras el valor sea distinto de
0

// Verificamos si se ingresaron valores (distintos de 0)

if (cant != 0) {

    val promedio = suma / cant      // Calculamos el promedio: suma total /
cantidad de valores

    print("El promedio de los valores ingresados es: $promedio")

} else

    print("No se ingresaron valores.") // Si no se ingresó nada válido,
mostramos este mensaje

}

```

Ejecución

```

Ingresar un valor (0 para finalizar):5
Ingresar un valor (0 para finalizar):70
Ingresar un valor (0 para finalizar):100
Ingresar un valor (0 para finalizar):66
Ingresar un valor (0 para finalizar):64
Ingresar un valor (0 para finalizar):0
El promedio de los valores ingresados es: 61

```

Ejercicio 44 - Piezas aptas

```

fun main(parametro: Array<String>) {
    var cant1 = 0      // Contador de piezas con peso > 10.2
    var cant2 = 0      // Contador de piezas aptas (entre 9.8 y 10.2 inclusive)
    var cant3 = 0      // Contador de piezas con peso < 9.8

    do {
        print("Ingresar el peso de la pieza (0 para finalizar):")
        val peso = readln().toDouble() // Se lee el peso ingresado por el
usuario

        // Clasificamos la pieza según el peso
        if (peso > 10.2)
            cant1++                  // Piezas con sobrepeso
    }
}
```

```

        else
            if (peso >= 9.8)
                cant2++           // Piezas aptas
            else
                if (peso > 0)
                    cant3++       // Piezas con bajo peso (excluyendo el 0)

    } while(peso != 0.0) // El ciclo continúa hasta que el peso sea 0

    // Mostramos los resultados
    println("Piezas aptas: $cant2")
    println("Piezas con un peso superior a 10.2: $cant1")
    println("Piezas con un peso inferior a 9.8: $cant3")

    // Calculamos el total de piezas procesadas (excluyendo el 0)
    val suma = cant1 + cant2 + cant3
    println("Cantidad total de piezas procesadas: $suma")
}

```

Ejecución

```

Ingrese el peso de la pieza (0 para finalizar):10.9
Ingrese el peso de la pieza (0 para finalizar):11
Ingrese el peso de la pieza (0 para finalizar):9.9
Ingrese el peso de la pieza (0 para finalizar):10.1
Ingrese el peso de la pieza (0 para finalizar):5
Ingrese el peso de la pieza (0 para finalizar):17
Ingrese el peso de la pieza (0 para finalizar):0
Piezas aptas: 2
Piezas con un peso superior a 10.2: 3
Piezas con un peso inferior a 9.8: 1
Cantidad total de piezas procesadas: 6

```

Problemas propuestos

Realizar un programa que acumule (sume) valores ingresados por teclado hasta ingresar el 9999 (no sumar dicho valor, indica que ha finalizado la carga). Imprimir el valor acumulado e informar si dicho valor es cero, mayor a cero o menor a cero.

En un banco se procesan datos de las cuentas corrientes de sus clientes. De cada cuenta corriente se conoce: número de cuenta y saldo actual. El ingreso de datos debe finalizar al ingresar un valor negativo en el número de cuenta.

Se pide confeccionar un programa que lea los datos de las cuentas corrientes e informe:

- a) De cada cuenta: número de cuenta y estado de la cuenta según su saldo, sabiendo que:

'Estado de la cuenta 'Acreedor' si el saldo es >0.

'Deudor' si el saldo es <0.

'Nulo' si el saldo es =0.

- b) La suma total de los saldos acreedores.

Ejercicio 45 - Suma hasta teclear 9999

```
fun main(parametro: Array<String>)    {  
    var suma = 0          // Variable acumuladora para la suma  
  
    do {  
        print("Ingrese un valor (finalizar con 9999):")  
        val valor = readln().toInt()    // Lee el valor ingresado  
  
        // Si el valor no es 9999, se acumula en 'suma'  
        if (valor != 9999)  
            suma += valor  
  
    } while (valor != 9999)    // El ciclo continúa mientras no se ingrese 9999  
  
    // Muestra el total acumulado  
    println("El valor acumulado es $suma")  
  
    // Determina el signo del total acumulado  
    if (suma == 0)  
        println("El valor acumulado es cero.")  
    else  
        if (suma > 0)  
            println("El valor acumulado es positivo.")  
        else  
            println("El valor acumulado es negativo")  
}
```

Ejecución

```
Ingrese un valor (finalizar con 9999):9000
Ingrese un valor (finalizar con 9999):90
Ingrese un valor (finalizar con 9999):80
Ingrese un valor (finalizar con 9999):10000
Ingrese un valor (finalizar con 9999):99999
Ingrese un valor (finalizar con 9999):9999
El valor acumulado es 119169
El valor acumulado es positivo.
```

Ejercicio 46 - Acreedor, deudor o nulo

```
fun main(parametro: Array<String>) {
    // Inicialización de la variable que acumulará los saldos acreedores
    var suma = 0.0

    // Bucle do-while para ingresar información de las cuentas
    do {
        // Solicitar al usuario que ingrese el número de cuenta
        print("Ingrese número de cuenta:")
        val cuenta = readln().toInt()

        // Verificar que la cuenta sea válida (mayor o igual a 0)
        if (cuenta >= 0) {
            // Solicitar al usuario que ingrese el saldo de la cuenta
            print("Ingrese saldo:")
            val saldo = readln().toDouble()

            // Clasificar el saldo según su valor
            if (saldo > 0) {
                // Si el saldo es positivo, es un saldo acreedor, y se acumula
                en la variable suma
                println("Saldo Acreedor.")
                suma += saldo
            } else if (saldo < 0) {
                // Si el saldo es negativo, es un saldo deudor
                println("Saldo Deudor.")
            }
        }
    } while (true)
}
```

```

        } else {
            // Si el saldo es igual a 0, es un saldo nulo
            println("Saldo Nulo.")
        }
    }

} while (cuenta >= 0) // El bucle continuará hasta que se ingrese una
cuenta negativa

// Al final, se imprime el total acumulado de saldos acreedores
println("Total de saldos Acreedores: $suma")
}

```

Ejecución

```

Ingrese número de cuenta:47473
Ingrese saldo:100
Saldo Acreedor.

Ingrese número de cuenta:38388
Ingrese saldo:20000
Saldo Acreedor.

Ingrese número de cuenta:19299
Ingrese saldo:-2000
Saldo Deudor.

Ingrese número de cuenta:48473
Ingrese saldo:0
Saldo Nulo.

Ingrese número de cuenta:-10
Total de saldos Acreedores: 20100.0

```

9. Estructura repetitiva for y expresiones de rango

Ejercicio 47 - Imprimir números hasta el cien

```

fun main(parametro: Array<String>) {
    // Un ciclo for que recorre los números del 1 al 100 (incluidos ambos
extremos)

    for(i in 1..100)
        println(i) // Imprime cada número en la consola
}

```

Ejecución

1	30	59	76
2	31	60	77
3	32	61	78
4	33	62	79
5	34	63	80
6	35	64	81
7	36	65	82
8	37	66	83
9	38	67	84
10	39	68	85
11	40	69	86
12	41	70	87
13	42	71	88
14	43	72	89
15	44	73	90
16	45	74	91
17	46	75	92
18	47	76	93
19	48	77	94
20	49	78	95
21	50	79	96
22	51	80	97
23	52	81	98
24	53	82	99
25	54	83	
26	55	84	
27	56	85	
28	57	86	
29	58	87	100

Ejercicio 48 - Imprimir números hasta el cien

```
fun main(parametro: Array<String>) {
    // Inicialización de la variable suma, que acumulará la suma de los
valores
    var suma = 0

    // Bucle for que se repite 10 veces, desde 1 hasta 10 (inclusive)
    for(i in 1..10) {
        // Solicitar al usuario que ingrese un valor
        print("Ingrese un valor:")

        // Leer el valor ingresado como un número entero
        val valor = readln().toInt()
```

```

    // Sumar el valor ingresado a la variable suma
    suma += valor
}

// Imprimir el total acumulado (la suma de los 10 valores)
println("La suma de los valores ingresados es $suma")

// Calcular el promedio dividiendo la suma entre 10
val promedio = suma / 10

// Imprimir el promedio de los valores
println("Su promedio es $promedio")
}

```

Ejecución

```

Ingrese un valor:3
Ingrese un valor:5
Ingrese un valor:7
Ingrese un valor:1
Ingrese un valor:0
Ingrese un valor:0
Ingrese un valor:2
Ingrese un valor:5
Ingrese un valor:100
Ingrese un valor:-9
La suma de los valores ingresados es 114
Su promedio es 11

```

Ejercicio 49 - Notas mayores o menores

```

fun main(parametro: Array<String>) {
    // Inicialización de las variables que llevarán el conteo de aprobados y
    reprobados

    var aprobados = 0
    var reprobados = 0

    // Bucle for que se repite 10 veces (para ingresar las notas de 10
alumnos)

    for(i in 1..10) {
        // Solicitar al usuario que ingrese la nota de un estudiante
    }
}

```

```

print("Ingrese nota:")

val nota = readln().toInt() // Lee la entrada como un número entero

// Si la nota es mayor o igual a 7, se considera aprobado
if (nota >= 7)
    aprobados++ // Incrementa el contador de aprobados
else
    reprobados++ // Si la nota es menor a 7, se considera reprobado
}

// Al finalizar el ciclo, se imprime la cantidad de alumnos aprobados
println("Cantidad de alumnos con notas mayores o iguales a 7: $aprobados")

// Se imprime la cantidad de alumnos reprobados (con notas menores a 7)
println("Cantidad de alumnos con notas menores a 7: $reprobados")
}

```

Ejecución

```

Ingrese nota:9
Ingrese nota:10
Ingrese nota:8
Ingrese nota:7
Ingrese nota:6
Ingrese nota:5
Ingrese nota:9
Ingrese nota:9
Ingrese nota:8
Ingrese nota:9
Cantidad de alumnos con notas mayores o iguales a 7: 8
Cantidad de alumnos con notas menores a 7: 2

```

Ejercicio 50 - Cantidad de múltiplos

```

fun main(parametro: Array<String>) {
    // Inicialización de las variables para contar múltiplos de 3, 5 y 9
    var mult3 = 0
    var mult5 = 0
    var mult9 = 0
}

```

```

// Bucle for que recorre los números del 1 al 10000
for(i in 1..10000) {
    // Si el número es divisible por 3, incrementa el contador de
    // múltiplos de 3
    if (i % 3 == 0)
        mult3++

    // Si el número es divisible por 5, incrementa el contador de
    // múltiplos de 5
    if (i % 5 == 0)
        mult5++

    // Si el número es divisible por 8, incrementa el contador de
    // múltiplos de 9
    if (i % 8 == 0)
        mult9++
}

// Imprime el número total de múltiplos de 3
println("Cantidad de múltiplos de 3: $mult3")

// Imprime el número total de múltiplos de 5
println("Cantidad de múltiplos de 5: $mult5")

// Imprime el número total de múltiplos de 9
println("Cantidad de múltiplos de 9: $mult9")
}

```

Ejecución

```

Cantidad de múltiplos de 3: 3333
Cantidad de múltiplos de 5: 2000
Cantidad de múltiplos de 9: 1250

```

Ejercicio 51 - Cantidad de pares

```

fun main(parametros: Array<String>) {
    var cant = 0 // Variable para contar cuántos números pares se ingresan
}

```

```

print("Cuantos valores ingresará para analizar:")
val cantidad = readln().toInt() // Se pide al usuario que indique cuántos números va a ingresar

// Bucle que se repite desde 1 hasta la cantidad indicada por el usuario
for(i in 1..cantidad) {
    print("Ingrese valor:")
    val valor = readln().toInt() // Se lee un número entero

    if (valor % 2 == 0) // Se verifica si el número es par (el residuo de dividir entre 2 es 0)
        cant++ // Si es par, se incrementa el contador
}

// Al finalizar el bucle, se muestra cuántos números pares se ingresaron
println("Cantidad de pares: $cant")
}

```

Ejecución

```

Cuantos valores ingresará para analizar:3
Ingrese valor:3
Ingrese valor:2
Ingrese valor:2
Cantidad de pares: 2

```

Problemas propuestos

Ejercicio 52 - Triángulos

```

fun main(parametro: Array<String>) {
    var cantidad = 0 // Contador para los triángulos cuya superficie sea mayor a 12

    print("Cuantos triángulos procesará:")
    val n = readln().toInt() // El usuario indica cuántos triángulos quiere analizar
}

```

```
for(i in 1..n) {  
    // Por cada triángulo se pide la base y la altura  
    print("Ingrese el valor de la base:")  
    val base = readln().toInt()  
  
    print("Ingrese el valor de la altura:")  
    val altura = readln().toInt()  
  
    // Se calcula la superficie del triángulo usando la fórmula: base *  
    altura / 2  
    val superficie = base * altura / 2  
    println("La superficie es de $superficie")  
  
    // Si la superficie es mayor a 12, se incrementa el contador  
    if (superficie > 12)  
        cantidad++  
    }  
  
    // Al finalizar el ciclo, se muestra cuántos triángulos superaron la  
    superficie de 12  
    print("La cantidad de triángulos con superficie superior a 12 son:  
$cantidad")  
}
```

Ejecución

```
Cuantos triángulos procesará:3
Ingrese el valor de la base:15
Ingrese el valor de la altura:15
La superficie es de 112
Ingrese el valor de la base:6
Ingrese el valor de la altura:5
La superficie es de 15
Ingrese el valor de la base:3
Ingrese el valor de la altura:6
La superficie es de 9
La cantidad de triángulos con superficie superior a 12 son: 2
```

Ejercicio 53 - Suma de los últimos cinco valores

```
fun main(parametro: Array<String>) {
    var suma = 0 // Variable acumuladora para sumar los últimos 5 valores
    ingresados

    for(i in 1..10) { // Se repite el proceso 10 veces
        print("Ingrese un valor entero:")

        val valor = readln().toInt() // Se lee un número entero desde el
        teclado

        if (i > 5) // Solo si estamos en las últimas 5 iteraciones (i = 6 a
        10)
            suma += valor // Se suma ese valor a la variable 'suma'

    }

    // Al terminar el bucle, se imprime la suma de los últimos 5 valores
    // ingresados
    print("La suma de los últimos 5 valores es: $suma");
}
```

Ejecución

```
Ingrese un valor entero:50
Ingrese un valor entero:10
Ingrese un valor entero:7
Ingrese un valor entero:6
Ingrese un valor entero:5
Ingrese un valor entero:9
Ingrese un valor entero:2
Ingrese un valor entero:0
Ingrese un valor entero:-20
Ingrese un valor entero:20
La suma de los últimos 5 valores es: 11
```

Ejercicio 54 - Tabla del cinco

```
fun main(parametro: Array<String>) { //Este código en Kotlin imprime la tabla
    del 5 desde el 5 hasta el 50.

    for(tabla5 in 5..50 step 5)
        println(tabla5)
}
```

Ejecución

```
5
10
15
20
25
30
35
40
45
50
```

Ejercicio 55 - Tabla de multiplicar

```
fun main(argumento: Array<String>) {
    print("Ingrese un valor entre 1 y 10:")
    val valor = readln().toInt() // Se lee el número ingresado por el usuario

    for(i in valor..valor * 12 step valor) // Se recorre desde ese valor hasta
    su múltiplo 12, aumentando de "valor" en "valor"
```

```
    println(i) // Se imprime cada múltiplo  
}
```

Ejecución

```
Ingrese un valor entre 1 y 10:6  
6  
12  
18  
24  
30  
36  
42  
48  
54  
60  
66  
72
```

Ejercicio 56 - Contador de Triángulos

```
fun main(argumento: Array<String>) {  
    var cant1 = 0 // Contador de triángulos equiláteros  
    var cant2 = 0 // Contador de triángulos isósceles  
    var cant3 = 0 // Contador de triángulos escalenos  
  
    print("Ingrese la cantidad de triángulos:")  
    val n = readln().toInt() // Se lee cuántos triángulos se van a procesar  
  
    for(i in 1..n) {  
        // Se ingresan los tres lados del triángulo  
        print("Ingrese lado 1:")  
        val lado1 = readln().toInt()  
        print("Ingrese lado 2:")  
        val lado2 = readln().toInt()  
        print("Ingrese lado 3:")  
        val lado3 = readln().toInt()  
  
        // Clasificación del triángulo según sus lados  
        if (lado1 == lado2 && lado1 == lado3) {  
            println("Es un triángulo equilátero.") // Todos los lados iguales
```

```

        cant1++
    } else if (lado1 == lado2 || lado1 == lado3 || lado2 == lado3) {
        println("Es un triángulo isósceles.") // Dos lados iguales
        cant2++
    } else {
        println("Es un triángulo escaleno.") // Todos los lados
diferentes
        cant3++
    }
}

// Se imprimen los resultados finales
println("Cantidad de triángulos equiláteros: $cant1")
println("Cantidad de triángulos isósceles: $cant2")
println("Cantidad de triángulos escalenos: $cant3")
}

```

Ejecución

```

Ingresar la cantidad de triángulos:3
Ingresar lado 1:10
Ingresar lado 2:13
Ingresar lado 3:10
Es un triángulo isósceles.
Ingresar lado 1:6
Ingresar lado 2:6
Ingresar lado 3:6
Es un triángulo equilátero.
Ingresar lado 1:13
Ingresar lado 2:10
Ingresar lado 3:12
Es un triángulo escaleno.
Cantidad de triángulos equiláteros: 1
Cantidad de triángulos isósceles: 1
Cantidad de triángulos escalenos: 1

```

Ejercicio 57 - Puntos en cuadrantes

```

fun main(parametro: Array<String>) {

```

```

var cant1 = 0 // Contador para puntos en el 1er cuadrante (x>0, y>0)
var cant2 = 0 // Contador para puntos en el 2do cuadrante (x<0, y>0)
var cant3 = 0 // Contador para puntos en el 3er cuadrante (x<0, y<0)
var cant4 = 0 // Contador para puntos en el 4to cuadrante (x>0, y<0)

// Se solicita cuántos puntos se van a ingresar
print("Cantidad de puntos a ingresar:")
val cantidad = readln().toInt()

// Se ejecuta un ciclo por cada punto
for(i in 1..cantidad) {
    // Se ingresan las coordenadas del punto
    print("Ingrese coordenada x:")
    val x = readln().toInt()
    print("Ingrese coordenada y:")
    val y = readln().toInt()

    // Clasificación del punto según su cuadrante
    if (x > 0 && y > 0)
        cant1++
    else if (x < 0 && y > 0)
        cant2++
    else if (x < 0 && y < 0)
        cant3++
    else if (x > 0 && y < 0)
        cant4++
}

// Mostrar los resultados
println("Cantidad de puntos en el primer cuadrante: $cant1")
println("Cantidad de puntos en el segundo cuadrante: $cant2")
println("Cantidad de puntos en el tercer cuadrante: $cant3")
println("Cantidad de puntos en el cuarto cuadrante: $cant4")
}

```

Ejecución

```
Cantidad de puntos a ingresar:3
Ingrese coordenada x:10
Ingrese coordenada y:-12
Ingrese coordenada x:0
Ingrese coordenada y:-11
Ingrese coordenada x:-10
Ingrese coordenada y:10
Cantidad de puntos en el primer cuadrante: 0
Cantidad de puntos en el segundo cuadrante: 1
Cantidad de puntos en el tercer cuadrante: 0
Cantidad de puntos en el cuarto cuadrante: 1
```

Ejercicio 58 - Cantidad de negativos, positivos, múltiplos de 15 y suma de pares

```
fun main(parametro: Array<String>) {
    var negativos = 0          // Contador para números negativos
    var positivos = 0          // Contador para números positivos
    var mult15 = 0              // Contador para múltiplos de 15
    var sumapares = 0          // Acumulador para sumar números pares

    // Bucle que se ejecuta 10 veces para ingresar 10 valores
    for(i in 1..10) {
        print("Ingrese valor:")
        val valor = readln().toInt() // Se lee el número ingresado por el
usuario

        // Se clasifica el número como negativo o positivo
        if (valor < 0)
            negativos++
        else
            if (valor > 0)
                positivos++

        // Se verifica si es múltiplo de 15
        if (valor % 15 == 0)
            mult15++
```

```
// Se verifica si es par y se suma en caso afirmativo  
if (valor % 2 == 0)  
    sumapares += valor  
  
// Resultados finales  
println("Cantidad de valores negativos: $negativos")  
println("Cantidad de valores positivos: $positivos")  
println("Cantidad de valores múltiplos de 15: $mult15")  
println("Suma de los valores pares: $sumapares")  
}
```

Ejecución

```
Ingrese valor:13  
Ingrese valor:-34  
Ingrese valor:13  
Ingrese valor:15  
Ingrese valor:3  
Ingrese valor:6  
Ingrese valor:60  
Ingrese valor:7  
Ingrese valor:66  
Ingrese valor:42  
Cantidad de valores negativos: 1  
Cantidad de valores positivos: 9  
Cantidad de valores múltiplos de 15: 2  
Suma de los valores pares: 140
```

10. Estructura condicional when

Ejercicio 59 - En qué eje se encuentra el punto

```
fun main(parametro: Array<String>) {  
    // Solicita coordenada x  
    print("Ingrese coordenada x del punto:")  
    val x = readln().toInt()  
  
    // Solicita coordenada y  
    print("Ingrese coordenada y del punto:")  
    val y = readln().toInt()  
  
    // Estructura condicional usando 'when'  
    when {  
        x > 0 && y > 0 -> println("Primer cuadrante")  
        x < 0 && y > 0 -> println("Segundo cuadrante")  
        x < 0 && y < 0 -> println("Tercer cuadrante")  
        x > 0 && y < 0 -> println("Cuarto cuadrante")  
        else -> println("El punto se encuentra en un eje") // Si x=0 o y=0  
    }  
}
```

Ejecución

```
Ingrese coordenada x del punto:13  
Ingrese coordenada y del punto:-10  
Cuarto cuadrante
```

Ejercicio 60 - Promocionado, regular o libre

```
fun main(parametros: Array<String>) {  
    // Entrada de tres notas  
    print("Ingrese primer nota:")  
    val nota1 = readln().toInt()  
    print("Ingrese segunda nota:")  
    val nota2 = readln().toInt()  
    print("Ingrese tercer nota:")
```

```

val nota3 = readln().toInt()

// Cálculo del promedio
val promedio = (notal + nota2 + nota3) / 3

// Clasificación según promedio
when {
    promedio >= 7 -> print("Promocionado")
    promedio >= 4 -> print("Regular")
    else -> print("Libre")
}
}

```

Ejecución

```

Ingrese primer nota:40
Ingrese segunda nota:50
Ingrese tercer nota:60
Promocionado

```

Ejercicio 61 - Piezas procesadas

```

fun main(parametro: Array<String>) {
    var cant1 = 0 // Cuenta cuántas piezas pesan más de 10.2 kg
    var cant2 = 0 // Cuenta cuántas piezas están dentro del rango apto (de
9.8 kg a 10.2 kg inclusive)
    var cant3 = 0 // Cuenta cuántas piezas pesan menos de 9.8 kg (pero
mayores a 0)

    do {
        print("Ingrese el peso de la pieza (0 para finalizar):")
        val peso = readln().toDouble() // Lee el peso como número decimal
        when {
            peso > 10.2 -> cant1++ // Piezas con sobrepeso
            peso >= 9.8 -> cant2++ // Piezas dentro del rango apto (9.8
a 10.2)
            peso > 0 -> cant3++ // Piezas con peso menor a 9.8 pero
mayor que 0
        }
    } while(peso != 0.0)
}

```

```

    println("Piezas aptas: $cant2") // Muestra cuántas piezas
están en el rango correcto

    println("Piezas con un peso superior a 10.2: $cant1") // Muestra cuántas
tienen sobrepeso

    println("Piezas con un peso inferior a 9.8: $cant3") // Muestra cuántas
están por debajo del mínimo

    val suma = cant1 + cant2 + cant3

    println("Cantidad total de piezas procesadas: $suma")
}

```

Ejecución

```

Ingresar el peso de la pieza (0 para finalizar):5
Ingresar el peso de la pieza (0 para finalizar):6
Ingresar el peso de la pieza (0 para finalizar):10
Ingresar el peso de la pieza (0 para finalizar):44
Ingresar el peso de la pieza (0 para finalizar):0
Piezas aptas: 1
Piezas con un peso superior a 10.2: 1
Piezas con un peso inferior a 9.8: 2
Cantidad total de piezas procesadas: 4

```

Ejercicio 62 - Sueldo bajo, medio o alto

```

fun main(parametro: Array<String>) {
    // Declaramos una variable llamada 'total' que almacenará la suma de los
suevos altos

    var total = 0

    // Creamos un bucle que se repite 10 veces, del 1 al 10 inclusive
    for(i in 1..10) {
        // Solicitamos al usuario que ingrese el sueldo del operario
        print("ingrese sueldo del operario:")

        // Leemos la entrada del usuario como texto y la convertimos a entero
        val sueldo = readln().toInt()

        // Usamos una expresión 'when' para clasificar el sueldo ingresado
        total += when {
            sueldo > 5000 -> {
                // Si el sueldo es mayor a 5000, es considerado "Sueldo alto"
                println("Sueldo alto")
            }
        }
    }
}

```

```

        // Se suma este sueldo al total porque es un sueldo alto
        sueldo
    }

    sueldo > 2000 -> {
        // Si el sueldo es mayor a 2000 pero menor o igual a 5000, es
        "Sueldo medio"

        println("Sueldo medio")

        // No se suma al total porque no es sueldo alto
        0
    }

    else -> {
        // Si el sueldo es menor o igual a 2000, se considera "Sueldo
        bajo"

        println("Sueldo bajo")

        // No se suma al total porque no es sueldo alto
        0
    }
}

}

// Finalmente, mostramos el gasto total en sueldos altos
println("Gastos totales en sueldos altos: $total")
}

```

Ejecución

```

ingrese sueldo del operario:3000
Sueldo medio
ingrese sueldo del operario:8000
Sueldo alto
ingrese sueldo del operario:1000
Sueldo bajo
ingrese sueldo del operario:1800
Sueldo bajo
ingrese sueldo del operario:2500
Sueldo medio
ingrese sueldo del operario:3000
Sueldo medio
ingrese sueldo del operario:3200
Sueldo medio

```

```

ingrese sueldo del operario:9000
Sueldo alto
ingrese sueldo del operario:2600
Sueldo medio
ingrese sueldo del operario:2000
Sueldo bajo
Gastos totales en sueldos altos: 17000

```

Problemas propuestos

Se ingresa por teclado un valor entero, mostrar una leyenda por pantalla que indique si el número es positivo, nulo o negativo.

Plantear una estructura que se repita 5 veces y dentro de la misma cargar 3 valores enteros. Acumular solo el mayor del cada lista de tres valores.

Realizar un programa que lea los lados de n triángulos, e informar:

- a) De cada uno de ellos, qué tipo de triángulo es: equilátero (tres lados iguales), isósceles (dos lados iguales), o escaleno (ningún lado igual)
- b) Cantidad de triángulos de cada tipo.

Ejercicio 63 - Cero, positivo o negativo

```
fun main(parametro: Array<String>) {  
    // Solicita al usuario que ingrese un número entero  
    print("Ingrese un valor entero:")  
  
    // Lee la entrada del usuario como texto y la convierte a un número entero  
    val valor = readln().toInt()  
  
    // Utiliza la estructura 'when' para verificar el valor ingresado  
    when {  
        valor == 0 ->  
            // Si el valor es igual a 0, se imprime este mensaje  
            println("Se ingresó el cero")  
        valor > 0 ->  
            // Si el valor es mayor que 0, se imprime este mensaje  
            println("Se ingresó un valor positivo")  
        else ->  
            // Si el valor no es ni 0 ni mayor que 0, entonces es negativo  
            println("Se ingresó un valor negativo")  
    }  
}
```

Ejecución

```
Ingrese un valor entero:-20
Se ingresó un valor negativo
```

Ejercicio 64 - Acumular el mayor de cada lista de 3 valores

```
fun main(parametro: Array<String>) {
    // Se declara una variable para acumular la suma de los valores mayores de
    // cada grupo de 3
    var suma = 0

    // Se repite el bloque 5 veces, es decir, se ingresarán 5 grupos de 3
    // valores
    for(i in 1..5) {
        // Sigue el usuario que ingrese el primer valor
        print("Ingrese primer valor:")
        val valor1 = readln().toInt()

        // Sigue el segundo valor
        print("Ingrese segundo valor:")
        val valor2 = readln().toInt()

        // Sigue el tercer valor
        print("Ingrese tercer valor:")
        val valor3 = readln().toInt()

        // Determina cuál de los tres valores es el mayor usando la expresión
        'when'
        // y lo suma a la variable 'suma'
        suma += when {
            valor1 > valor2 && valor1 > valor3 -> valor1 // Si valor1 es mayor
            que los otros dos
            valor2 > valor3 -> valor2 // Si valor2 es mayor que valor3 (y
            menor que valor1)
            else -> valor3 // En cualquier otro caso, valor3 es el mayor
        }
    }
}
```

```

    // Imprime el resultado acumulado de los valores mayores de cada grupo de
3

    println("El valor acumulado de los mayores de cada lista de 3 valores es
: $suma")
}

```

Ejecución

```

Ingresé primer valor:80
Ingresé segundo valor:19
Ingresé tercer valor:77
Ingresé primer valor:13
Ingresé segundo valor:44
Ingresé tercer valor:21
Ingresé primer valor:0
Ingresé segundo valor:12
Ingresé tercer valor:33
Ingresé primer valor:55
Ingresé segundo valor:32
Ingresé tercer valor:2
Ingresé primer valor:2
Ingresé segundo valor:3
Ingresé tercer valor:56
El valor acumulado de los mayores de cada lista de 3 valores es : 268

```

Ejercicio 65 - Cantidad de triángulos

```

fun main(argumento: Array<String>) {

    // Variables contadoras para cada tipo de triángulo

    var cant1 = 0 // Equiláteros
    var cant2 = 0 // Isósceles
    var cant3 = 0 // Escalenos

    // Solicita al usuario cuántos triángulos quiere ingresar
    print("Ingresé la cantidad de triángulos:")
    val n = readln().toInt()

    // Bucle que se repite 'n' veces (una vez por cada triángulo)
    for(i in 1..n) {
        // Solicita los tres lados del triángulo
        print("Ingresé lado 1:")
    }
}

```

```

val lado1 = readln().toInt()
print("Ingrese lado 2:")
val lado2 = readln().toInt()
print("Ingrese lado 3:")
val lado3 = readln().toInt()

// Clasifica el tipo de triángulo según los lados ingresados
when {
    // Si los tres lados son iguales, es equilátero
    lado1 == lado2 && lado1 == lado3 -> {
        println("Es un triángulo equilátero.")
        cant1++ // Aumenta el contador de equiláteros
    }
    // Si dos lados son iguales, es isósceles
    lado1 == lado2 || lado1 == lado3 || lado2 == lado3 -> {
        println("Es un triángulo isósceles.")
        cant2++ // Aumenta el contador de isósceles
    }
    // Si todos los lados son distintos, es escaleno
    else -> {
        println("Es un triángulo escaleno.")
        cant3++ // Aumenta el contador de escalenos
    }
}

// Imprime la cantidad total de cada tipo de triángulo
println("Cantidad de triángulos equilateros: $cant1")
println("Cantidad de triángulos isósceles: $cant2")
println("Cantidad de triángulos escalenos: $cant3")
}

```

Ejecución

```
Ingrese la cantidad de triángulos:2
Ingrese lado 1:23
Ingrese lado 2:23
Ingrese lado 3:23
Es un triángulo equilátero.

Ingrese lado 1:14
Ingrese lado 2:12
Ingrese lado 3:10
Es un triángulo escaleno.

Cantidad de triángulos equiláteros: 1
Cantidad de triángulos isósceles: 0
Cantidad de triángulos escalenos: 1
```

Ejercicio 66 - Número en texto

```
fun main(parametro: Array<String>) {
    // Solicita al usuario que ingrese un número entero entre 1 y 5
    print("Ingrese un valor entero entre 1 y 5:")

    // Lee la entrada del usuario como texto y la convierte a número entero
    val valor = readln().toInt()

    // Utiliza una estructura 'when' para comparar el valor ingresado
    when (valor) {
        1 -> print("uno")           // Si el valor es 1, imprime "uno"
        2 -> print("dos")           // Si el valor es 2, imprime "dos"
        3 -> print("tres")          // Si el valor es 3, imprime "tres"
        4 -> print("cuatro")         // Si el valor es 4, imprime "cuatro"
        5 -> print("cinco")          // Si el valor es 5, imprime "cinco"
        else -> print("valor fuera de rango") // Si el valor no está entre 1 y
5
    }
}
```

Ejecución

```
Ingrese un valor entero entre 1 y 5:4
cuatro
```

Ejercicio 67 - Cuántos dígitos tiene el número

```
fun main(parametro: Array<String>) {  
    // Solicita al usuario que ingrese un número entero positivo entre 1 y  
    99999  
    print("Ingrese un valor entero positivo comprendido entre 1 y 99999:")  
  
    // Lee la entrada del usuario y la convierte a un entero  
    val valor = readln().toInt()  
  
    // Usa la estructura 'when' para determinar cuántos dígitos tiene el  
    // número ingresado  
  
    when (valor) {  
        in 1..9 -> print("Tiene 1 dígito") // Si el número está entre  
        1 y 9  
        in 10..99 -> print("Tiene 2 dígitos") // Si está entre 10 y 99  
        in 100..999 -> print("Tiene 3 dígitos") // Si está entre 100 y 999  
        in 1000..9999 -> print("Tiene 4 dígitos") // Si está entre 1000 y  
        9999  
        in 10000..99999 -> print("Tiene 5 dígitos") // Si está entre 10000 y  
        99999  
        else -> print("No se encuentra comprendido en el rango indicado") //  
        Si no está en el rango  
    }  
}
```

Ejecución

```
Ingrese un valor entero positivo comprendido entre 1 y 99999:666  
Tiene 3 dígitos
```

Ejercicio 68 - Uno, cinco, diez o cero

```
fun main(parametro: Array<String>) {  
    // Contadores para diferentes tipos de valores  
    var cant1 = 0 // Contador para la cantidad de ceros ingresados  
    var cant2 = 0 // Contador para la cantidad de 1, 5 o 10 ingresados  
  
    // Repite el ingreso 10 veces  
    for(i in 1..10) {
```

```

// Pide al usuario que ingrese un valor entero
print("Ingrese un valor entero:")

val valor = readln().toInt()

// Evalúa el valor ingresado usando 'when'
when (valor) {
    0 -> cant1++ // Si es 0, incrementa cant1
    1, 5, 10 -> cant2++ // Si es 1, 5 o 10, incrementa cant2
}
}

// Imprime los resultados
println("Cantidad de números 0 ingresados: $cant1")
println("Cantidad de números 1, 5 o 10 ingresados: $cant2")
}

```

Ejecución

```

Ingrese un valor entero:10
Ingrese un valor entero:0
Ingrese un valor entero:6
Ingrese un valor entero:12
Ingrese un valor entero:3
Ingrese un valor entero:5
Ingrese un valor entero:4
Ingrese un valor entero:2
Ingrese un valor entero:1
Ingrese un valor entero:0
Cantidad de números 0 ingresados: 2
Cantidad de números 1, 5 o 10 ingresados: 3

```

Problema propuesto

Realizar la carga de la cantidad de hijos de 10 familias. Contar cuantos tienen 0,1,2 o más hijos. Imprimir dichos contadores.

Ejercicio 69 - Hijos en la familia

```

fun main(parametro: Array<String>) {
    // Variables contadoras para clasificar las familias según la cantidad de hijos
    var conta1 = 0 // Contador para familias con 0 hijos
    var conta2 = 0 // Contador para familias con 1 hijo
}

```

```

var conta3 = 0 // Contador para familias con 2 hijos
var conta4 = 0 // Contador para familias con más de 2 hijos

// Repite el ingreso de datos 10 veces (una por familia)
for(i in 1..10) {
    // Solicita al usuario que ingrese la cantidad de hijos para una
familia
    print("Ingrese la cantidad de hijos de la familia:")
    val cantidad = readln().toInt()

    // Clasifica la cantidad de hijos usando 'when' y actualiza el
contador correspondiente
    when (cantidad) {
        0 -> conta1++           // Si la familia tiene 0 hijos
        1 -> conta2++           // Si la familia tiene 1 hijo
        2 -> conta3++           // Si la familia tiene 2 hijos
        else -> conta4++        // Si tiene más de 2 hijos
    }
}

// Muestra los resultados acumulados
println("Cantidad de familias con 0 hijos: $conta1")
println("Cantidad de familias con 1 hijos: $conta2")
println("Cantidad de familias con 2 hijos: $conta3")
println("Cantidad de familias con más de 2 hijos: $conta4")
}

```

Ejecución

```

Ingrese la cantidad de hijos de la familia:3
Ingrese la cantidad de hijos de la familia:2
Ingrese la cantidad de hijos de la familia:1
Ingrese la cantidad de hijos de la familia:5
Ingrese la cantidad de hijos de la familia:4
Ingrese la cantidad de hijos de la familia:2
Ingrese la cantidad de hijos de la familia:1
Ingrese la cantidad de hijos de la familia:0
Ingrese la cantidad de hijos de la familia:2
Ingrese la cantidad de hijos de la familia:3
Cantidad de familias con 0 hijos: 1
Cantidad de familias con 1 hijos: 2
Cantidad de familias con 2 hijos: 3
Cantidad de familias con más de 2 hijos: 4

```

11. Concepto de funciones

Ejercicio 70 - Función suma de valores

```
// Función que muestra la presentación del programa

fun presentacion() {
    println("Programa que permite cargar dos valores por teclado.") // Mensaje introductorio
    println("Efectua la suma de los valores") // Explica qué hace el programa
    println("Muestra el resultado de la suma") // Describe el resultado
    println("*****") // Línea decorativa
}

// Función que carga dos valores, los suma y muestra el resultado

fun cargarSumar() {
    print("Ingrese el primer valor:") // Sigue el primer número
    val valor1 = readln().toInt() // Lo lee y convierte a entero
    print("Ingrese el segundo valor:") // Sigue el segundo número
    val valor2 = readln().toInt() // Lo lee y convierte a entero
    val suma = valor1 + valor2 // Realiza la suma
    println("La suma de los dos valores es: $suma") // Muestra el resultado
}

// Función que imprime un mensaje de despedida

fun finalizacion() {
    println("*****") // Línea decorativa
    println("Gracias por utilizar este programa") // Mensaje final
}

// Función principal: punto de entrada del programa

fun main(parametro: Array<String>) {
    presentacion() // Llama a la función de presentación
    cargarSumar() // Llama a la función que suma los valores
    finalizacion() // Llama a la función de despedida
}
```

```
}
```

Ejecución

```
Programa que permite cargar dos valores por teclado.  
Efectua la suma de los valores  
Muestra el resultado de la suma  
*****  
Ingrese el primer valor:4  
Ingrese el segundo valor:66  
La suma de los dos valores es: 70  
*****  
Gracias por utilizar este programa
```

Ejercicio 71 - Función suma de valores cinco veces

```
// Función que carga dos valores por teclado, los suma y muestra el resultado  
  
fun cargarSuma() {  
    print("Ingrese el primer valor:") // Solicita el primer número al  
    usuario  
    val valor1 = readln().toInt() // Lee el primer valor y lo  
    convierte a entero  
    print("Ingrese el segundo valor:") // Solicita el segundo número  
    val valor2 = readln().toInt() // Lee el segundo valor y lo  
    convierte a entero  
    val suma = valor1 + valor2 // Realiza la suma de los dos  
    valores  
    println("La suma de los dos valores es: $suma") // Muestra el resultado  
    de la suma  
}  
  
// Función que imprime una línea de separación decorativa  
  
fun separacion() {  
    println("*****") // Imprime una línea para  
    separar visualmente las operaciones  
}  
  
// Función principal donde inicia la ejecución del programa  
  
fun main(parametro: Array<String>) {  
    // Bucle que repite 5 veces el ingreso y suma de valores  
    for (i in 1..5) {  
        cargarSuma() // Llama a la función que suma dos valores
```

```

        separacion()      // Llama a la función que imprime la línea de
separación
    }
}

```

Ejecución

```

Ingrese el primer valor:5
Ingrese el segundo valor:24
La suma de los dos valores es: 29
*****
Ingrese el primer valor:1
Ingrese el segundo valor:1
La suma de los dos valores es: 2
*****
Ingrese el primer valor:2
Ingrese el segundo valor:6
La suma de los dos valores es: 8
*****
Ingrese el primer valor:6
Ingrese el segundo valor:6
La suma de los dos valores es: 12
*****
Ingrese el primer valor:6
Ingrese el segundo valor:7
La suma de los dos valores es: 13
*****

```

Problemas propuestos

Desarrollar un programa con dos funciones. La primer solicite el ingreso de un entero y muestre el cuadrado de dicho valor. La segunda que solicite la carga de dos valores y muestre el producto de los mismos. Llamar desde la main a ambas funciones.

Desarrollar una función que solicite la carga de tres valores y muestre el menor. Desde la función main del programa llamar 2 veces a dicha función (sin utilizar una estructura repetitiva)

Ejercicio 72 - Cuadrado y producto de valores

```

// Función que calcula el cuadrado de un número
fun calculaCuadrado() {
    print("Ingrese un entero:")                                // Pide al usuario que ingrese
un número entero

    val valor = readln().toInt()                             // Lee el valor y lo
    convierte a entero

```

```

        val cuadrado = valor * valor                      // Calcula el cuadrado del
valor ingresado

        println("El cuadrado es $cuadrado")                // Muestra el resultado del
cuadrado

    }

// Función que calcula el producto de dos valores

fun calcularProducto() {

    print("Ingrese primer valor:")                     // Solicita el primer valor al
usuario

    val valor1 = readln().toInt()                      // Lee el primer valor

    print("Ingrese segundo valor:")                   // Solicita el segundo valor

    val valor2 = readln().toInt()                      // Lee el segundo valor

    val producto = valor1 * valor2                  // Calcula el producto de los
dos valores

    println("El producto de los valores es: $producto") // Muestra el
resultado del producto

}

// Función principal que llama a las otras dos funciones

fun main(parametro: Array<String>) {
    calculaCuadrado()      // Llama a la función para calcular el cuadrado de
un número

    calcularProducto()     // Llama a la función para calcular el producto de
dos números
}

```

Ejecución

```

Ingrese un entero:3
El cuadrado es 9
Ingrese primer valor:13
Ingrese segundo valor:6
El producto de los valores es: 78

```

Ejercicio 73 - Función menor valor

```

// Función que determina el menor de tres valores ingresados

fun menorValor() {
    print("Ingrese primer valor:")                  // Solicita el primer número
al usuario

```

```

    val valor1 = readln().toInt()                      // Lee el primer valor como
entero

    print("Ingrese segundo valor:")                  // Solicita el segundo número
    val valor2 = readln().toInt()                      // Lee el segundo valor
    print("Ingrese tercer valor:")                  // Solicita el tercer número
    val valor3 = readln().toInt()                      // Lee el tercer valor

    print("Menor de los tres:")                     // Muestra mensaje previo al
resultado

    when {
        // Compara y determina cuál es el menor valor entre los tres
        valor1 < valor2 && valor1 < valor3 -> println(valor1)
        valor2 < valor3 -> println(valor2)
        else -> println(valor3)
    }
}

// Función principal que llama dos veces a la función menorValor
fun main(parametro: Array<String>) {
    menorValor()          // Llama la primera vez a la función para pedir 3
números y mostrar el menor
    menorValor()          // Llama una segunda vez, repitiendo el mismo proceso
}

```

Ejecución

```

Ingresé primer valor:5
Ingresé segundo valor:4
Ingresé tercer valor:3
Menor de los tres:3
Ingresé primer valor:66
Ingresé segundo valor:77
Ingresé tercer valor:65
Menor de los tres:65

```

12. Funciones: parámetros

Ejercicio 74 - Función que calcula la suma de los valores ingresados

```
// Función que muestra un mensaje enmarcado entre líneas de asteriscos
fun mostrarMensaje(mensaje: String) {
    println("*****") // Línea superior decorativa
    println(mensaje) // Muestra el mensaje recibido como parámetro
    println("*****") // Línea inferior decorativa
}

// Función que solicita dos números, los suma y muestra el resultado
fun cargarSumar() {
    print("Ingrese el primer valor:") // Pide al usuario el primer número
    val valor1 = readln().toInt() // Lee y convierte a entero el primer valor
    print("Ingrese el segundo valor:") // Pide al usuario el segundo número
    val valor2 = readln().toInt() // Lee y convierte a entero el segundo valor
    val suma = valor1 + valor2 // Realiza la suma de los dos valores
    println("La suma de los dos valores es: $suma") // Muestra el resultado
}

// Función principal del programa
fun main(parametro: Array<String>) {
    // Muestra un mensaje introductorio usando la función mostrarMensaje
    mostrarMensaje("El programa calcula la suma de dos valores ingresados por teclado.")

    // Llama a la función que pide dos números y muestra su suma
    cargarSumar()

    // Muestra un mensaje de despedida usando la función mostrarMensaje
}
```

```
        mostrarMensaje("Gracias por utilizar este programa")
    }
```

Ejecución

```
*****
El programa calcula la suma de dos valores ingresados por teclado.
*****
Ingrese el primer valor:30
Ingrese el segundo valor:21
La suma de los dos valores es: 51
*****
Gracias por utilizar este programa
*****
```

Ejercicio 75 - Función mayor de tres valores

```
// Función que recibe tres valores enteros y muestra cuál es el mayor
fun mostrarMayor(v1: Int, v2: Int, v3: Int) {
    print("Mayor:") // Imprime el texto antes de mostrar el valor mayor
    if (v1 > v2 && v1 > v3) // Si v1 es mayor que v2 y v3
        println(v1) // Imprime v1
    else
        if (v2 > v3) // Si v1 no es el mayor, compara v2 y v3
            print(v2) // Imprime v2 si es mayor que v3
        else
            print(v3) // Si no, imprime v3 porque es el mayor
}

// Función principal
fun main(parametro: Array<String>) {
    // Solicita al usuario que ingrese tres valores
    print("Ingrese primer valor:")
    val valor1 = readln().toInt()

    print("Ingrese segundo valor:")
    val valor2 = readln().toInt()

    print("Ingrese tercer valor:")
```

```

    val valor3 = readln().toInt()

    // Llama a la función para determinar y mostrar el mayor de los tres
    mostrarMayor(valor1, valor2, valor3)
}

```

Ejecución

```

Ingrese primer valor:3
Ingrese segundo valor:6
Ingrese tercer valor:1
Mayor:6

```

Ejercicio 76 - Función perímetro o superficie

```

// Función para calcular y mostrar el perímetro de un cuadrado
fun mostrarPerimetro(lado: Int) {
    val perimetro = lado * 4 // El perímetro de un cuadrado es 4 veces el lado
    println("El perímetro es $perimetro") // Muestra el resultado
}

// Función para calcular y mostrar la superficie (área) de un cuadrado
fun mostrarSuperficie(lado: Int) {
    val superficie = lado * lado // La superficie de un cuadrado es lado al cuadrado
    println("La superficie es $superficie") // Muestra el resultado
}

// Función principal del programa
fun main(parametro: Array<String>) {
    // Pide al usuario que ingrese el valor del lado del cuadrado
    print("Ingrese el valor del lado de un cuadrado:")
    val la = readln().toInt()

    // Pregunta qué desea calcular: perímetro o superficie
    print("Quiere calcular el perímetro o la superficie [ingresar texto: perímetro/superficie]")
    var respuesta = readln()
}

```

```

    // Según la respuesta del usuario, se llama a la función correspondiente
    when (respuesta) {
        "perimetro" -> mostrarPerimetro(la)          // Llama a la función de
perímetro

        "superficie" -> mostrarSuperficie(la)        // Llama a la función de
superficie
    }
}

```

Ejecución

```

Ingrese el valor del lado de un cuadrado:4
Quiere calcular el perimetro o la superficie [ingresar texto: perimetro/superficie]superficie
La superficie es 16

```

Problemas propuestos

En la función main solicitar que se ingrese una clave dos veces por teclado.

Desarrollar una función que reciba dos String como parametros y muestre un mensaje si las dos claves ingresadas son iguales o distintas.

Confeccionar una función que reciba tres enteros y los muestre ordenados de menor a mayor. En la función main solicitar la carga de 3 enteros por teclado y proceder a llamar a la primer función definida.

Ejercicio 77 - Función verificar claves

```

// Función que compara dos claves ingresadas por el usuario
fun verificarClaves(clave1: String, clave2: String) {
    // Compara si ambas claves son exactamente iguales
    if (clave1 == clave2)

        println("Se ingresaron las dos veces la misma clave") // Muestra
mensaje si son iguales

    else

        print("No se ingresó las dos veces con el mismo valor") // Muestra
mensaje si son diferentes
}

// Función principal del programa
fun main(parametro: Array<String>) {
    // Solicita al usuario que ingrese la primera clave
    print("Ingrese primer clave:")
}

```

```

val clave1 = readln()

// Solicita al usuario que repita la misma clave
print("Repita el ingreso de la misma clave;")
val clave2 = readln()

// Llama a la función verificarClaves con las dos entradas del usuario
verificarClaves(clave1, clave2)
}

```

Ejecución

```

Ingrese primer clave;4242
Repita el ingreso de la misma clave;4242
Se ingresaron las dos veces la misma clave

```

Ejercicio 78 - Función ordenar menor a mayor

```

// Función que ordena tres números enteros de menor a mayor y los muestra
fun ordenarMayorMenor(valor1: Int, valor2: Int, valor3: Int) {
    when {
        // Si valor1 es el menor
        valor1 < valor2 && valor1 < valor3 -> {
            if (valor2 < valor3)
                println("$valor1 $valor2 $valor3")
            else
                println("$valor1 $valor3 $valor2")
        }
        // Si valor2 es el menor
        valor2 < valor3 -> {
            if (valor1 < valor3)
                println("$valor2 $valor1 $valor3")
            else
                println("$valor2 $valor3 $valor1")
        }
        // Si valor3 es el menor
        else -> {
            if (valor1 < valor2)

```

```

        println("$valor3 $valor1 $valor2")
    else
        println("$valor3 $valor2 $valor1")
    }
}

// Función principal: pide al usuario tres números e invoca la función para
// ordenarlos

fun main(parametros: Array<String>) {
    print("Ingrese primer valor: ")
    val valor1 = readln().toInt()

    print("Ingrese segundo valor: ")
    val valor2 = readln().toInt()

    print("Ingrese tercer valor: ")
    val valor3 = readln().toInt()

    // Llama a la función para mostrar los valores ordenados
    ordenarMayorMenor(valor1, valor2, valor3)
}

```

Ejecución

```

Ingrese primer valor: 90
Ingrese segundo valor: 45
Ingrese tercer valor: 60
45 60 90

```

13. Funciones: con retorno de datos

Ejercicio 79 - Función que calcula y retorna la superficie de un cuadrado

```
// Función que calcula y retorna la superficie de un cuadrado dado su lado
fun retornarSuperficie(lado: Int): Int {
    val sup = lado * lado // Superficie = lado × lado
    return sup           // Retorna el resultado
}

fun main(parametro: Array<String>) {
    print("Ingrese el valor del lado del cuadrado: ") // Corrección de
    'cuadrado' a 'cuadrado'
    val la = readln().toInt() // Lee el valor del lado desde teclado

    val superficie = retornarSuperficie(la) // Llama a la función para
    calcular la superficie

    println("La superficie del cuadrado es $superficie") // Muestra el
    resultado
}
```

Ejercicio 80 - Función que retorna el mayor de dos valores

```
// Retorna el mayor entre dos valores
fun retornarMayor(v1: Int, v2: Int): Int {
    if (v1 > v2)
        return v1
    else
        return v2
}

// Función principal que pide dos valores y muestra el mayor
fun main(parametro: Array<String>) {
    print("Ingrese el primer valor:")
    val valor1 = readLine()!!.toInt()
    print("Ingrese el segundo valor:")
    val valor2 = readLine()!!.toInt()
```

```
    println("El mayor entre $valor1 y $valor2 es ${retornarMayor(valor1,  
valor2)}")  
}
```

Ejecución

```
Ingrese el primer valor:44  
Ingrese el segundo valor:55  
El mayor entre 44 y 55 es 55
```

Ejercicio 81 - Función nombre más largo

```
// Función que retorna la cantidad de caracteres de un nombre  
  
fun largo(nombre: String): Int {  
    return nombre.length  
}  
  
fun main(parametro: Array<String>) {  
    print("Ingrese un nombre: ")  
    val nombre1 = readln()  
  
    print("Ingrese otro nombre: ")  
    val nombre2 = readln()  
  
    // Comparamos las longitudes usando la función largo()  
    if (largo(nombre1) == largo(nombre2)) {  
        println("Los nombres: $nombre1 y $nombre2 tienen la misma cantidad de  
caracteres")  
    } else if (largo(nombre1) > largo(nombre2)) {  
        println("$nombre1 es más largo")  
    } else {  
        println("$nombre2 es más largo")  
    }  
}
```

Ejecución

```
Ingrese un nombre: Karla
Ingrese otro nombre: Zafiro
Zafiro es más largo
```

Problemas propuestos

Elaborar una función que reciba tres enteros y nos retorne el valor promedio de los mismos.
Elaborar una función que nos retorne el perímetro de un cuadrado pasando como parámetros el valor del lado.

Confeccionar una función que calcule la superficie de un rectángulo y la retorne, la función recibe como parámetros los valores de dos de sus lados:

```
fun retornarSuperficie(lado1: Int,lado2: Int): Int
```

En la función main del programa cargar los lados de dos rectángulos y luego mostrar cual de los dos tiene una superficie mayor.

Ejercicio 82 - Función promedio de tres números

```
// Función que recibe tres enteros y devuelve el promedio como un número decimal (Double)

fun retornarPromedio(v1: Int, v2: Int, v3: Int): Double {
    val promedio = (v1 + v2 + v3) / 3.0 // Usamos 3.0 para que la división sea con decimales
    return promedio // Devolvemos el promedio como Double
}

fun main(parametro: Array<String>) {
    // Se piden tres valores al usuario
    print("Ingrese primer valor:")
    val valor1 = readln().toInt()

    print("Ingrese segundo valor:")
    val valor2 = readln().toInt()

    print("Ingrese tercer valor:")
    val valor3 = readln().toInt()
```

```

    // Se llama a la función retornarPromedio y se imprime el resultado con
    decimales

    println("Valor promedio de los tres números ingresados es
    ${retornarPromedio(valor1, valor2, valor3)}")
}

```

Ejecución

```

Ingrese primer valor:5
Ingrese segundo valor:3
Ingrese tercer valor:6
Valor promedio de los tres números ingresados es 4.666666666666667

```

Ejercicio 83 - Función perímetro de un cuadrado

```

// Función que recibe un parámetro 'lado' de tipo Int y devuelve el perímetro
// del cuadrado

fun retornarPerímetro(lado: Int): Int {
    // Calcula el perímetro del cuadrado, multiplicando el lado por 4 (porque
    // un cuadrado tiene 4 lados iguales)
    val perímetro = lado * 4

    // Devuelve el valor calculado del perímetro
    return perímetro
}

fun main(parametro: Array<String>) {
    // Pide al usuario que ingrese el valor del lado del cuadrado
    print("Ingrese el lado del cuadrado:")

    // Lee el valor ingresado por el usuario y lo convierte a un entero (Int)
    val lado = readln().toInt()

    // Llama a la función retornarPerímetro pasándole el valor del lado, y
    // luego imprime el resultado
    // Muestra el perímetro del cuadrado en la consola
    print("El perímetro es: ${retornarPerímetro(lado)}")
}

```

Ejecución

```
Ingrese el lado del cuadrado:6  
El perimetro es: 24
```

Ejercicio 84 - Función rectángulo de mayor superficie

```
// Función que calcula la superficie de un rectángulo, recibiendo como  
parámetros los lados del rectángulo  
  
fun retornarSuperficie(lado1: Int, lado2: Int): Int {  
    // Retorna la superficie, que es el producto de los dos lados (lado1 y  
    lado2)  
    return lado1 * lado2  
}  
  
  
fun main(parametro: Array<String>) {  
    // Mensaje para indicar que el usuario ingresará datos para el primer  
    rectángulo  
    println("Primer rectangulo")  
  
    // Sigue el mismo patrón que el ejercicio anterior, pero con un cambio:  
    // Solicita al usuario que ingrese el valor del lado menor del primer  
    rectángulo  
    print("Ingrese lado menor del rectangulo:")  
    val lado1 = readln().toInt()    // Convierte la entrada del usuario a un  
    entero  
  
    // Sigue el mismo patrón que el ejercicio anterior, pero con un cambio:  
    // Solicita al usuario que ingrese el valor del lado mayor del primer  
    rectángulo  
    print("Ingrese lado mayor del rectangulo:")  
    val lado2 = readln().toInt()    // Convierte la entrada del usuario a un  
    entero  
  
    // Mensaje para indicar que el usuario ingresará datos para el segundo  
    rectángulo  
    println("Segundo rectangulo")  
  
    // Sigue el mismo patrón que el ejercicio anterior, pero con un cambio:  
    // Solicita al usuario que ingrese el valor del lado menor del segundo  
    rectángulo  
    print("Ingrese lado menor del rectangulo:")  
    val lado3 = readln().toInt()    // Convierte la entrada del usuario a un  
    entero
```

```

    // Solicita al usuario que ingrese el valor del lado mayor del segundo
    // rectángulo

    print("Ingrese lado mayor del rectangulo:")

    val lado4 = readln().toInt()    // Convierte la entrada del usuario a un
    entero

    // Compara las superficies de los dos rectángulos

    if (retornarSuperficie(lado1, lado2) == retornarSuperficie(lado3, lado4))
    {

        // Si las superficies son iguales, se imprime este mensaje

        print("Los dos rectangulos tienen la misma superficie")

    } else {

        // Si la superficie del primer rectángulo es mayor que la del segundo

        if (retornarSuperficie(lado1, lado2) > retornarSuperficie(lado3,
lado4)) {

            // Imprime que el primer rectángulo tiene una superficie mayor

            print("El primer rectangulo tiene una superficie mayor")

        } else {

            // Si la superficie del segundo rectángulo es mayor que la del
            primero

            // Imprime que el segundo rectángulo tiene una superficie mayor

            print("El segundo rectangulo tiene una superficie mayor")

        }

    }

}

```

Ejecución

```

Primer rectangulo
Ingrese lado menor del rectangulo:6
Ingrese lado mayor del rectangulo:7
Segundo rectangulo
Ingrese lado menor del rectangulo:12
Ingrese lado mayor del rectangulo:15
El segundo rectangulo tiene una superficie mayor

```

14. Funciones: con retorno de datos

Ejercicio 85 - Función superficie de un cuadrado

```
// Función que devuelve la superficie de un cuadrado.

// Usamos la expresión de función de una sola línea (función de expresión)
que calcula el área.

fun retornarSuperficie(lado: Int) = lado * lado

fun main(parametro: Array<String>) {
    // Imprime el mensaje solicitando al usuario que ingrese el valor del lado
    // del cuadrado
    print("Ingrese el valor del lado del cuadrado:")

    // Lee la entrada del usuario, la convierte a un entero y la guarda en la
    variable 'la'
    val la = readln().toInt()

    // Imprime el resultado, llamando a la función 'retornarSuperficie' con el
    valor de 'la'
    // El resultado es calculado y mostrado en la misma línea con el uso de la
    interpolación de cadenas
    println("La superficie del cuadrado es ${retornarSuperficie(la)}")
}
```

Ejecución

```
Ingrese el valor del lado del cuadrado:4
La superficie del cuadrado es 16
```

Ejercicio 86 - Función retornarMayor

```
// Definición de la función 'retornarMayor', que toma dos enteros como
parámetros

// y devuelve el mayor de los dos. Utiliza una expresión 'if' como una forma
compacta de escribir el código.

fun retornarMayor(v1: Int, v2: Int) = if (v1 > v2) v1 else v2

fun main(parametro: Array<String>) {
    // Solicita al usuario que ingrese el primer valor y lo almacena en la
    variable 'valor1'
    print("Ingrese el primer valor:")
```

```

val valor1 = readln().toInt() // Convierte la entrada a un entero

// Solicita al usuario que ingrese el segundo valor y lo almacena en la
variable 'valor2'

print("Ingrese el segundo valor:")

val valor2 = readln().toInt() // Convierte la entrada a un entero

// Llama a la función 'retornarMayor' pasando los valores 'valor1' y
'valor2' como argumentos

// Luego, imprime el mayor de los dos valores usando la interpolación de
cadenas en Kotlin.

println("El mayor entre $valor1 y $valor2 es ${retornarMayor(valor1,
valor2)}")

}

```

Ejecución

```

Ingrese el primer valor:6
Ingrese el segundo valor:13
El mayor entre 6 y 13 es 13

```

Ejercicio 87 - Función convertirCastelano

```

// Definición de la función 'convertirCastelano', que toma un valor entero
como parámetro

// y devuelve una cadena que representa ese valor en español. Si el valor no
está en el rango de 1 a 5, devuelve "error".

fun convertirCastelano(valor: Int) = when (valor) {

    1 -> "uno"        // Si 'valor' es 1, devuelve "uno"
    2 -> "dos"        // Si 'valor' es 2, devuelve "dos"
    3 -> "tres"       // Si 'valor' es 3, devuelve "tres"
    4 -> "cuatro"     // Si 'valor' es 4, devuelve "cuatro"
    5 -> "cinco"      // Si 'valor' es 5, devuelve "cinco"
    else -> "error"   // Si 'valor' no es 1, 2, 3, 4 o 5, devuelve "error"
}

fun main(parametro: Array<String>) {

    // Un bucle 'for' que recorre los números del 1 al 6 (inclusive)
    for(i in 1..6) {
        // Para cada número en el rango, llama a la función
        'convertirCastelano'
    }
}

```

```

        // y imprime el resultado (el valor convertido a español o "error").
        println(convertirCastellano(i))
    }
}

```

Ejecución

```

uno
dos
tres
cuatro
cinco
error

```

Problemas propuestos

Utilizar una única expresión en las funciones pedidas en estos problemas

Elaborar una función que reciba tres enteros y nos retorne el valor promedio de los mismos.

Elaborar una función que nos retorne el perímetro de un cuadrado pasando como parámetros el valor del lado.

Confeccionar una función que calcule la superficie de un rectángulo y la retorne, la función recibe como parámetros los valores de dos de sus lados:

fun retornarSuperficie(lado1: Int,lado2: Int): Int

En la función main del programa cargar los lados de dos rectángulos y luego mostrar cual de los dos tiene una superficie mayor.

Confeccionar una función que le envíemos como parámetro un String y nos retorne la cantidad de caracteres que tiene. En la función main solicitar la carga de dos nombres por teclado y llamar a la función dos veces. Imprimir en la main cual de las dos palabras tiene más caracteres.

Ejercicio 88 - Función retornarPromedio

```

// Función que recibe tres enteros y retorna su promedio entero.
// Se utiliza una expresión de una sola línea (expresión abreviada).

fun retornarPromedio(v1: Int, v2: Int, v3: Int) = (v1 + v2 + v3) / 3

fun main(parametro: Array<String>) {
    // Solicita al usuario que ingrese el primer valor
    print("Ingrese primer valor:")
}

```

```

    val valor1 = readln().toInt() // Lee la entrada del usuario y la
    convierte a Int

    // Solicita el segundo valor
    print("Ingrese segundo valor:")
    val valor2 = readln().toInt()

    // Solicita el tercer valor
    print("Ingrese tercer valor:")
    val valor3 = readln().toInt()

    // Llama a la función 'retornarPromedio' con los tres valores y muestra el
    resultado
    println("Valor promedio de los tres números ingresados es
    ${retornarPromedio(valor1, valor2, valor3)}")
}

```

Ejecución

```

Ingrese primer valor:77
Ingrese segundo valor:16
Ingrese tercer valor:60
Valor promedio de los tres números ingresados es 51

```

Ejercicio 89 - Función retornarPerímetro

```

// Función que recibe el valor de un lado de un cuadrado y retorna su
perímetro.

// El perímetro de un cuadrado es 4 veces el valor de un lado.

fun retornarPerimetro(lado: Int) = lado * 4

fun main(parametro: Array<String>) {
    // Solicita al usuario que ingrese el valor de un lado del cuadrado.
    print("Ingrese el lado del cuadrado:")

    // Lee la entrada del usuario desde la consola y la convierte a un número
    entero (Int).

    val lado = readln().toInt()

    // Llama a la función 'retornarPerimetro' pasándole el lado ingresado y
    muestra el resultado.
}

```

```

    // El resultado se incrusta directamente dentro del texto usando ${...}
    print("El perimetro es: ${retornarPerimetro(lado)}")
}


```

Ejecución

```

Ingrese el lado del cuadrado:6
El perimetro es: 24

```

Ejercicio 90 - Función retornarSuperficie

```

// Función que calcula la superficie (área) de un rectángulo.
// Se recibe la base (lado1) y la altura (lado2), y se devuelve su
multiplicación.

fun retornarSuperficie(lado1: Int, lado2: Int) = lado1 * lado2

fun main(parametro: Array<String>) {
    // Sección para ingresar los datos del primer rectángulo
    println("Primer rectangulo")
    print("Ingrese lado menor del rectangulo:") // Solicita el lado menor
    val lado1 = readln().toInt() // Lee y convierte la entrada
a entero

    print("Ingrese lado mayor del rectangulo:") // Solicita el lado mayor
    val lado2 = readln().toInt() // Lee y convierte la entrada
a entero

    // Sección para ingresar los datos del segundo rectángulo
    println("Segundo rectangulo")
    print("Ingrese lado menor del rectangulo:") // Solicita el lado menor
    val lado3 = readln().toInt() // Lee y convierte la entrada
a entero

    print("Ingrese lado mayor del rectangulo:") // Solicita el lado mayor
    val lado4 = readln().toInt() // Lee y convierte la entrada
a entero

    // Compara las superficies de ambos rectángulos usando la función
retornarSuperficie

    if (retornarSuperficie(lado1, lado2) == retornarSuperficie(lado3, lado4))
        print("Los dos rectangulos tienen la misma superficie")
}


```

```

        else if (retornarSuperficie(lado1, lado2) > retornarSuperficie(lado3,
lado4))

            print("El primer rectangulo tiene una superficie mayor")

        else

            print("El segundo rectangulo tiene una superficie mayor")

}

```

Ejecución

```

Primer rectangulo
Ingrese lado menor del rectangulo:10
Ingrese lado mayor del rectangulo:13
Segundo rectangulo
Ingrese lado menor del rectangulo:13
Ingrese lado mayor del rectangulo:14
El segundo rectangulo tiene una superficie mayor

```

Ejercicio 91 - Función largo

```

// Función que recibe un nombre (cadena de texto) y retorna su longitud
(cantidad de caracteres)

fun largo(nombre: String) = nombre.length

fun main(parametro: Array<String>) {
    // Solicita al usuario que ingrese el primer nombre
    print("Ingrese un nombre:")
    val nombre1 = readln() // Lee el primer nombre ingresado por el usuario

    // Solicita al usuario que ingrese el segundo nombre
    print("Ingrese otro nombre:")
    val nombre2 = readln() // Lee el segundo nombre ingresado por el usuario

    // Compara la longitud de ambos nombres
    if (largo(nombre1) == largo(nombre2))

        // Si tienen la misma cantidad de caracteres, imprime un mensaje indicando
        eso

        print("Los nombres: $nombre1 y $nombre2 tienen la misma cantidad de
caracteres")

    else

        // Si no tienen la misma longitud, determina cuál es más largo
        if (largo(nombre1) > largo(nombre2))

```

```

    // Si el primer nombre es más largo, lo indica
    print("$nombre1 es más largo")

    else
        // Si el segundo nombre es más largo, lo indica
        print("$nombre2 es más largo")
}

```

Ejecución

```

Ingrese un nombre:Omar
Ingrese otro nombre:Lia
Omar es más largo

```

15. Funciones: con parámetros con valor por defecto

Ejercicio 92 - Función subrayado

```

// Función que imprime un título y debajo lo "subraya" con un carácter dado.
// Parámetros:
//   - titulo: el texto que se mostrará como título.
//   - caracter: el símbolo con el que se subrayará el título (por defecto "*").
fun tituloSubrayado(titulo: String, caracter: String = "*") {
    // Imprime el título tal cual
    println(titulo)

    // Repite la impresión del 'caracter' tantas veces como la longitud del
    // título
    // para crear una línea de subrayado de igual longitud.
    for (i in 1..titulo.length)
        print(caracter)

    // Mueve el cursor a la siguiente línea tras el subrayado
    println()
}

fun main(parametro: Array<String>) {
    // Llama a la función usando el carácter predeterminado ("*")
    tituloSubrayado("Sistema de Administracion")

    // Llama a la función especificando un guion ("") como carácter de
    // subrayado
}

```

```
    tituloSubrayado("Ventas", "-")
}
```

Ejecución

```
Sistema de Administracion
*****
Ventas
-----
```

Problemas propuestos

Confeccionar una función que reciba entre 2 y 5 enteros. La misma nos debe retornar la suma de dichos valores. Debe tener tres parámetros por defecto.

Ejercicio 93 - Función subrrayado

```
// Definimos una función llamada 'sumar' que acepta hasta 5 números enteros.
// Los últimos tres parámetros (v3, v4 y v5) tienen valores por defecto de 0.
// Esto significa que si no se pasan esos valores al llamar la función, se
utilizarán como 0.

fun sumar(v1: Int, v2: Int, v3: Int = 0, v4: Int = 0, v5: Int = 0) = v1 + v2
+ v3 + v4 + v5

fun main(parametro: Array<String>) {
    // Se llama a la función pasando solo dos argumentos, por lo tanto v3, v4
y v5 son 0

    // Resultado: 5 + 6 + 0 + 0 + 0 = 11
    println("La suma de 5 + 6 es ${sumar(5,6)}")

    // Se pasan tres argumentos, v4 y v5 son 0 por defecto
    // Resultado: 1 + 2 + 3 + 0 + 0 = 6
    println("La suma de 1 + 2 + 3 es ${sumar(1,2,3)}")

    // Se pasan los cinco argumentos, se suman todos
    // Resultado: 1 + 2 + 3 + 4 + 5 = 15
    print("La suma de 1 + 2 + 3 + 4 + 5 es ${sumar(1,2,3,4,5)}")
}
```

Ejecución

```
La suma de 5 + 6 es 11
La suma de 1 + 2 + 3 es 6
La suma de 1 + 2 + 3 + 4 + 5 es 15
```

16. Funciones: Llamada a la función con argumentos nombrados

Ejercicio 94 - Función sueldo

```
// Definimos una función llamada 'calcularSueldo' que recibe:
// - un nombre (String)
// - un costo por hora (Double)
// - una cantidad de horas trabajadas (Int)

fun calcularSueldo(nombre: String, costoHora: Double, cantidadHoras: Int) {
    // Se calcula el sueldo multiplicando las horas trabajadas por el costo
    // por hora

    val sueldo = costoHora * cantidadHoras

    // Se imprime un mensaje con los datos del trabajador y el sueldo
    // calculado

    println("$nombre trabajó $cantidadHoras horas, se le paga por hora
$costoHora por lo tanto le corresponde un sueldo de $sueldo")
}

fun main(parametro: Array<String>) {
    // Llamada a la función pasando los parámetros por posición
    calcularSueldo("juan", 10.5, 120)

    // Llamada a la función usando nombres de parámetros (puedes ponerlos en
    // cualquier orden)

    calcularSueldo(costoHora = 12.0, cantidadHoras = 40, nombre = "ana")

    // Otra llamada usando nombre de parámetros en distinto orden
    calcularSueldo(cantidadHoras = 90, nombre = "luis", costoHora = 7.25)
}
```

Ejecución

```
juan trabajó 120 horas, se le paga por hora 10.5 por lo tanto le corresponde un sueldo de 1260.0
ana trabajó 40 horas, se le paga por hora 12.0 por lo tanto le corresponde un sueldo de 480.0
luis trabajó 90 horas, se le paga por hora 7.25 por lo tanto le corresponde un sueldo de 652.5
```

Problemas propuestos

Elaborar una función que muestre la tabla de multiplicar del valor que le enviamos como parámetro. Definir un segundo parámetro llamado termino que por defecto almacene el valor 10. Se deben mostrar tantos términos de la tabla de multiplicar como lo indica el segundo parámetro.

Llamar a la función desde la main con argumentos nombrados.

Ejercicio 95 - Función tabla

```
// Función que imprime la tabla de multiplicar de un número hasta cierta cantidad de términos.

// Parámetros:
// - numero: el número base cuya tabla queremos mostrar.
// - terminos: la cantidad de términos a mostrar (por defecto es 10).

fun tabla(numero: Int, terminos: Int = 10) {
    // Usamos un bucle que inicia en 'numero' y se incrementa en pasos de 'numero'

    // hasta llegar a numero * terminos (inclusive si coincide).
    for (i in numero..numero * terminos step numero)
        println(i)
}

fun main(parametro: Array<String>) {
    println("Tabla del 3")
    tabla(3) // Imprime la tabla del 3 hasta 10 términos: 3, 6, 9, ..., 30

    println("Tabla del 3 con 5 terminos")
    tabla(3, 5) // Imprime la tabla del 3 hasta 5 términos: 3, 6, 9, 12, 15

    println("Tabla del 3 con 20 terminos")
    tabla(terminos = 20, numero = 3) // Uso de argumentos nombrados. Imprime del 3 al 60 en pasos de 3
}
```

Ejecución

```
Tabla del 3
3
6
9
12
15
18
21
24
27
30
Tabla del 3 con 5 terminos
3
6
9
12
15
Tabla del 3 con 20 terminos
3
6
9
12
15
18
21
24
27
30
33
36
39
42
45
48
51
54
57
60
```

17. Funciones: internas o locales

Ejercicio 96 - Función multiplo2y5

```
// Esta función principal analiza cuántos de los números ingresados son
múltiplos de 2 y de 5

fun multiplos2y5() {

    // Función interna que determina si un número es múltiplo de otro
    // Devuelve true si el número es divisible sin residuo (modulo % == 0)
    fun multiplo(numero: Int, valor: Int) = numero % valor == 0

    // Contadores para múltiplos de 2 y 5
    var mult2 = 0
    var mult5 = 0

    // Se repite 10 veces para pedir al usuario un número en cada iteración
    for(i in 1..10) {
```

```

print("Ingrese valor:")

    val valor = readln().toInt() // Lee un número del teclado y lo
    convierte a entero

        // Si el número es múltiplo de 2, incrementa el contador
        correspondiente

        if (multiplo(valor, 2))
            mult2++

        // Si el número es múltiplo de 5, incrementa el otro contador

        if (multiplo(valor, 5))
            mult5++

    }

    // Al finalizar las 10 entradas, muestra cuántos múltiplos de 2 y de 5
    hubo

    println("Cantidad de múltiplos de 2 : $mult2")
    println("Cantidad de múltiplos de 5 : $mult5")
}

// Función principal desde donde se llama a la función multiplos2y5()

fun main(parametro: Array<String>) {
    multiplos2y5()
}

```

Ejecución

```

Ingrese valor:4
Ingrese valor:3
Ingrese valor:6
Ingrese valor:10
Ingrese valor:9
Ingrese valor:0
Ingrese valor:-4
Ingrese valor:-13
Ingrese valor:2
Ingrese valor:15
Cantidad de múltiplos de 2 : 6
Cantidad de múltiplos de 5 : 3

```

Problemas propuestos

Confeccionar una función que permita cargar dos enteros y nos muestre el mayor de ellos. Realizar esta actividad con 5 pares de valores.

Implementar una función interna que retorne el mayor de dos enteros.

Ejercicio 97 - Función Mayor

```
// Función que se encarga de mostrar el mayor entre dos números, repitiéndose
5 veces

fun mostrarMayor() {

    // Función interna que compara dos números y retorna el mayor
    fun mayor(x1: Int, x2: Int) = if (x1 > x2) x1 else x2

    // Bucle que se repite 5 veces
    for(i in 1..5) {
        // Solicita el primer valor al usuario
        print("Ingrese primer valor:")
        val valor1 = readln().toInt() // Lee y convierte a entero

        // Solicita el segundo valor al usuario
        print("Ingrese segundo valor:")
        val valor2 = readln().toInt() // Lee y convierte a entero

        // Muestra cuál de los dos valores es mayor
        println("El mayor entre $valor1 y $valor2 es ${mayor(valor1,
valor2)}")
    }
}

// Función principal que llama a mostrarMayor
fun main(parametro: Array<String>) {
    mostrarMayor()
}
```

Ejecución

```
Ingrese primer valor:10
Ingrese segundo valor:6
El mayor entre 10 y 6 es 10
Ingrese primer valor:13
Ingrese segundo valor:20
El mayor entre 13 y 20 es 20
Ingrese primer valor:4
Ingrese segundo valor:5
El mayor entre 4 y 5 es 5
Ingrese primer valor:6
Ingrese segundo valor:4
El mayor entre 6 y 4 es 6
Ingrese primer valor:6
Ingrese segundo valor:6
El mayor entre 6 y 6 es 6
```

18. Funciones: conceptos

Ejercicio 98 - Sueldos función

```
fun main(parametro: Array<String>) {
    // Declaración de un arreglo (array) de enteros llamado "sueldos"
    val sueldos: IntArray

    // Inicialización del arreglo con 5 elementos (todos inicialmente en 0)
    sueldos = IntArray(5)

    // Carga de valores al arreglo: se piden 5 sueldos al usuario
    for(i in 0..4) {
        print("Ingrese sueldo:")
        sueldos[i] = readln().toInt() // Se convierte la entrada a entero y
        se guarda en la posición i
    }

    // Impresión de los sueldos ingresados
    for(i in 0..4) {
        println(sueldos[i]) // Muestra el sueldo guardado en la posición i
    }
}
```

Ejecución

```
Ingrese sueldo:1000
Ingrese sueldo:2000
Ingrese sueldo:3000
Ingrese sueldo:1000
Ingrese sueldo:0
1000
2000
3000
1000
0
```

Ejercicio 99 - Promedio de alturas

```
// Calcula el promedio de alturas e indica cuántas están por encima o por
debajo

fun main(parametro: Array<String>) {
    val alturas = FloatArray(5)
    var suma = 0f
    for(i in 0 until alturas.size) {
        print("Ingrese la altura:")
        alturas[i] = readLine()!!.toFloat()
        suma += alturas[i]
    }
    val promedio = suma / alturas.size
    println("Altura promedio: $promedio")
    var altos = 0
    var bajos = 0
    for(i in 0 until alturas.size)
        if (alturas[i] > promedio)
            altos++
        else
            bajos++
    println("Cantidad de personas más altas que el promedio: $altos")
    println("Cantidad de personas más bajas que el promedio: $bajos")
}
```

Ejecución

```
Ingrese la altura:1.70
Ingrese la altura:1.81
Ingrese la altura:1.60
Ingrese la altura:1.54
Ingrese la altura:1.00
Altura promedio: 1.53
Cantidad de personas más altas que el promedio: 4
Cantidad de personas más bajas que el promedio: 1
```

Ejercicio 100 - Arreglo menor a mayor

```
// Verifica si los elementos de un arreglo están ordenados de menor a mayor
fun main(parametro: Array<String>) {
    val arreglo = IntArray(10)
    for(i in 0 until arreglo.size) {
        print("Ingrese elemento:")
        arreglo[i] = readLine()!!.toInt()
    }
    var ordenado = true
    for(i in 0 until arreglo.size - 1)
        if (arreglo[i+1] < arreglo[i])
            ordenado = false
    if (ordenado)
        print("Los elementos están ordenados de menor a mayor")
    else
        print("Los elementos no están ordenados de menor a mayor")
}
```

Ejecución

```
Ingrese elemento:90
Ingrese elemento:100
Ingrese elemento:150
Ingrese elemento:200
Ingrese elemento:300
Ingrese elemento:1000
Ingrese elemento:1200
Ingrese elemento:1300
Ingrese elemento:3000
Ingrese elemento:3300
Los elementos están ordenados de menor a mayor
```

Ejercicio 101 - Arreglo primero y último

```
// Carga 10 elementos en un arreglo e imprime el primero y el último

fun main(parametro: Array<String>) {
    val arreglo = IntArray(10)
    for(i in arreglo.indices) {
        print("Ingrese elemento:")
        arreglo[i] = readLine()!!.toInt()
    }
    println("Primera componente del arreglo ${arreglo[0]}")
    println("Última componente del arreglo ${arreglo[arreglo.lastIndex]}")
}
```

Ejecución

```
Ingrese elemento:10
Ingrese elemento:14
Ingrese elemento:1
Ingrese elemento:3
Ingrese elemento:5
Ingrese elemento:6
Ingrese elemento:4
Ingrese elemento:2
Ingrese elemento:9
Ingrese elemento:10
Primera componente del arreglo 10
Última componente del arreglo 10
```

Ejercicio 102 - Diez elementos del arreglo

```
// Carga 10 elementos en un arreglo e imprime todos sus valores

fun main(parametro: Array<String>) {
    val arreglo = IntArray(10)
    for(i in arreglo.indices) {
        print("Ingrese elemento:")
        arreglo[i] = readLine()!!.toInt()
    }
}
```

```
for(elemento in arreglo)
    println(elemento)
}
```

Ejecución

```
Ingrese elemento:100
Ingrese elemento:4
Ingrese elemento:5
Ingrese elemento:223
Ingrese elemento:66
Ingrese elemento:4
Ingrese elemento:67
Ingrese elemento:88
Ingrese elemento:6
Ingrese elemento:32
100
4
5
223
66
4
67
88
6
32
```

Problemas propuestos

Desarrollar un programa que permita ingresar un arreglo de 8 elementos enteros, e informe:

El valor acumulado de todos los elementos.

El valor acumulado de los elementos que sean mayores a 36.

Cantidad de valores mayores a 50.

(Definir dos for, en el primero realizar la carga y en el segundo proceder a analizar cada elemento)

Realizar un programa que pida la carga de dos arreglos numéricos enteros de 4 elementos. Obtener la suma de los dos arreglos elemento a elemento, dicho resultado guardarlo en un tercer arreglo del mismo tamaño.

Ejercicio 103 - Valor acumulado arreglo

```
fun main(parametro: Array<String>) {
    // Se declara un arreglo de 8 enteros con valores iniciales en 0
    val arreglo = IntArray(8)

    // Ciclo para llenar el arreglo con valores ingresados por el usuario
```

```

for(i in arreglo.indices) {
    print("Ingrese elemento:")           // Solicita al usuario que ingrese
un número

    arreglo[i] = readln().toInt()        // Lee la entrada, la convierte a
entero y la guarda en el arreglo

}

// Se inicializan las variables para los cálculos

var suma = 0                         // Suma total de todos los elementos del arreglo
var sumaMayor36 = 0                   // Suma de elementos mayores a 36
var cantMayor50 = 0                   // Conteo de elementos mayores a 50

// Se recorre el arreglo para calcular las sumas y contar elementos

for(elemento in arreglo) {
    suma += elemento                  // Acumula el total de los
elementos

    if (elemento > 30)
        sumaMayor36 += elemento      // Si el elemento es mayor a 30,
se suma a 'sumaMayor36'

    if (elemento > 50)
        cantMayor50++              // Si el elemento es mayor a 50,
se incrementa el contador
}

// Se muestran los resultados en consola

println("Valor acumulado del arreglo: $suma")
println("Valor acumulado de los elementos mayores a 36: $sumaMayor36")
println("Cantidad de elementos mayores a 50: $cantMayor50")
}

```

Ejecución

```
Ingrese elemento:3
Ingrese elemento:4
Ingrese elemento:5
Ingrese elemento:1
Ingrese elemento:2
Ingrese elemento:8
Ingrese elemento:7
Ingrese elemento:5
Valor acumulado del arreglo: 35
Valor acumulado de los elementos mayores a 36: 0
Cantidad de elementos mayores a 50: 0
```

Ejercicio 104 - Arreglo resultante

```
fun main(parametro: Array<String>) {
    // Se crean dos arreglos de 4 enteros cada uno
    val arreglo1 = IntArray(4)
    val arreglo2 = IntArray(4)

    println("Carga del primer arreglo")
    // Se llena el primer arreglo con valores ingresados por el usuario
    for(i in arreglo1.indices) {
        print("Ingrese elemento: ")
        arreglo1[i] = readln().toInt() // Se convierte la entrada a entero
    }

    println("Carga del segundo arreglo")
    // Se llena el segundo arreglo con valores ingresados por el usuario
    for(i in arreglo2.indices) {
        print("Ingrese elemento: ")
        arreglo2[i] = readln().toInt()
    }

    // Se crea un tercer arreglo para guardar la suma de los otros dos
    val arregloSuma = IntArray(4)

    // Se suman los elementos en la misma posición de ambos arreglos
    for(i in arregloSuma.indices)
```

```

        arregloSuma[i] = arreglo1[i] + arreglo2[i]

    // Se imprime el arreglo resultante
    println("Arreglo resultante")
    for(elemento in arregloSuma)
        println(elemento)
}

```

Ejecución

```

Carga del primer arreglo
Ingrese elemento: 12
Ingrese elemento: 1
Ingrese elemento: 4
Ingrese elemento: 5
Carga del segundo arreglo
Ingrese elemento: 3
Ingrese elemento: 8
Ingrese elemento: 34
Ingrese elemento: 20
Arreglo resultante
15
9
38
25

```

19. Funciones: parámetros y retorno de datos tipo arreglo

Ejercicio 105 - Elementos de un arreglo

```

// Carga y muestra los elementos de un arreglo
fun cargar(arreglo: IntArray) {
    for(i in arreglo.indices) {
        print("Ingrese elemento:")
        arreglo[i] = readLine()!!.toInt()
    }
}

fun imprimir(arreglo: IntArray) {
    for(elemento in arreglo)
        println(elemento)
}

```

```

}

fun main(parametro: Array<String>) {
    val arre = IntArray(5)
    cargar(arre)
    imprimir(arre)
}

```

Ejecución

```

Ingrese elemento:13
Ingrese elemento:12
Ingrese elemento:1
Ingrese elemento:9
Ingrese elemento:13
13
12
1
9
13

```

Ejercicio 106 - Imprime sueldos de un arreglo

```

// Carga una cantidad variable de sueldos en un arreglo y los imprime
fun cargar(): IntArray {
    print("Cuantos sueldos quiere cargar:")
    val cantidad = readLine()!!.toInt()
    val sueldos = IntArray(cantidad)
    for(i in sueldos.indices) {
        print("Ingrese elemento:")
        sueldos[i] = readLine()!!.toInt()
    }
    return sueldos
}

fun imprimir(sueldos: IntArray) {
    println("Listado de todos los sueldos")
    for(sueldo in sueldos)
        println(sueldo)
}

fun main(parametro: Array<String>) {
    val sueldos = cargar()
    imprimir(sueldos)
}

```

```
}
```

Ejecución

```
Cuantos sueldos quiere cargar:3
Ingrese elemento:1000
Ingrese elemento:3000
Ingrese elemento:200
Listado de todos los sueldos
1000
3000
200
```

Problemas propuestos

Desarrollar un programa que permita ingresar un arreglo de n elementos, ingresar n por teclado.

Elaborar dos funciones una donde se lo cree y cargue al arreglo y otra que sume todos sus elementos y retorne dicho valor a la main donde se lo imprima.

Cargar un arreglo de n elementos. Imprimir el menor elemento y un mensaje si se repite dentro del arreglo.

Ejercicio 107 - Arreglar cargar

```
// Función para cargar un arreglo con valores ingresados por el usuario
fun cargar(): IntArray {
    print("¿Cuántos elementos tendrá el arreglo?: ")
    val cantidad = readln().toInt() // Se lee la cantidad de elementos
    val arreglo = IntArray(cantidad) // Se crea el arreglo con esa cantidad

    for (i in arreglo.indices) {
        print("Ingrese elemento: ")
        arreglo[i] = readln().toInt() // Se llena el arreglo con valores ingresados
    }
    return arreglo // Se devuelve el arreglo completo
}

// Función para imprimir todos los elementos del arreglo
```

```

fun imprimir(arreglo: IntArray) {
    println("Listado completo del arreglo:")
    for (elemento in arreglo)
        println(elemento) // Imprime cada elemento
    uno por uno
}

// Función para sumar los elementos del arreglo
fun sumar(arreglo: IntArray): Int {
    var suma = 0
    for (elemento in arreglo)
        suma += elemento // Acumula cada número en
    la variable 'suma'
    return suma // Devuelve la suma total
}

// Función principal que une todo
fun main(parametro: Array<String>) {
    val arreglo = cargar() // Carga el arreglo con datos del usuario
    imprimir(arreglo) // Imprime el contenido del arreglo
    println("La suma de todos sus elementos es: ${sumar(arreglo)}") // Muestra la suma de los elementos
}

```

Ejecución

```

¿Cuántos elementos tendrá el arreglo?: 2
Ingrese elemento: 123
Ingrese elemento: 2222
Listado completo del arreglo:
123
2222
La suma de todos sus elementos es: 2345

```

Ejercicio 108 - Arreglar cargar

```

// Función para cargar un arreglo con datos del usuario
fun cargar(): IntArray {
    print("Cuántos elementos tendrá el arreglo: ")

```

```

    val cantidad = readln().toInt()                                // Se pide la cantidad
de elementos

    val arreglo = IntArray(cantidad)                               // Se crea el arreglo con
ese tamaño

    // Se llena el arreglo con valores del usuario

    for (i in arreglo.indices) {
        print("Ingrese elemento: ")
        arreglo[i] = readln().toInt()
    }

    return arreglo                                              // Se devuelve el
arreglo lleno
}

// Función para imprimir todos los elementos del arreglo

fun imprimir(arreglo: IntArray) {
    println("Listado completo del arreglo:")
    for (elemento in arreglo)
        println(elemento)
}

// Función para encontrar el elemento mayor del arreglo

fun mayor(arreglo: IntArray): Int {
    var may = arreglo[0]                                         // Se toma el primer
elemento como el mayor inicial

    for (elemento in arreglo)
        if (elemento > may)
            may = elemento                                      // Se actualiza si se
encuentra un número mayor

    return may
}

// Función para saber si un número aparece más de una vez

fun repite(arreglo: IntArray, buscar: Int): Boolean {
    var cant = 0
    for (elemento in arreglo)
        if (elemento == buscar)
            cant++                                            // Se cuenta cuántas
veces aparece el valor
}

```

```

        return cant > 1                                // Devuelve true si
aparece más de una vez

}

// Función principal
fun main(parametro: Array<String>) {
    val arreglo = cargar()                          // Carga el arreglo
desde consola

    imprimir(arreglo)                            // Muestra todos los
elementos

    val mayorElemento = mayor(arreglo)           // Busca el mayor
    println("El elemento mayor es $mayorElemento") // Imprime el mayor

    // Verifica si el mayor se repite
    if (repite(arreglo, mayorElemento))
        println("Se repite el mayor en el arreglo")
    else
        println("No se repite el mayor en el arreglo")
}

```

Ejecución

```

Cuántos elementos tendrá el arreglo: 3
Ingrese elemento: 12
Ingrese elemento: 44
Ingrese elemento: 122
Listado completo del arreglo:
12
44
122
El elemento mayor es 122
No se repite el mayor en el arreglo

```

20. POO -Conceptos de programación orientada a objetos

Ejercicio 109 - Clase persona

```
// Clase Persona con inicialización, impresión y verificación de mayoría de edad

class Persona {

    var nombre: String = ""

    var edad: Int = 0

    fun inicializar(nombre: String, edad: Int) {
        this.nombre = nombre
        this.edad = edad
    }

    fun imprimir() {
        println("Nombre: $nombre y tiene una edad de $edad")
    }

    fun esMayorEdad() {
        if (edad >= 18)
            println("Es mayor de edad $nombre")
        else
            println("No es mayor de edad $nombre")
    }
}

fun main(parametro: Array<String>) {
    val personal = Persona()
    personal.inicializar("Juan", 12)
    personal.imprimir()
    personal.esMayorEdad()
    val persona2 = Persona()
    persona2.inicializar("Ana", 50)
    persona2.imprimir()
    persona2.esMayorEdad()
}
```

Ejecución

```
Nombre: Juan y tiene una edad de 12
No es mayor de edad Juan
Nombre: Ana y tiene una edad de 50
Es mayor de edad Ana
```

Ejercicio 110 - Clase triangulo

```
// Clase Triangulo con métodos para inicializar, verificar lado mayor y si es
equilátero

class Triangulo {

    var lado1: Int = 0
    var lado2: Int = 0
    var lado3: Int = 0

    fun inicializar() {
        print("Ingrese lado 1:")
        lado1 = readLine()!!.toInt()
        print("Ingrese lado 2:")
        lado2 = readLine()!!.toInt()
        print("Ingrese lado 3:")
        lado3 = readLine()!!.toInt()
    }

    fun ladoMayor() {
        print("Lado mayor:")
        when {
            lado1 > lado2 && lado1 > lado3 -> println(lado1)
            lado2 > lado3 -> println(lado2)
            else -> println(lado3)
        }
    }

    fun esEquilatero() {
        if (lado1 == lado2 && lado1 == lado3)
            print("Es un triángulo equilátero")
        else
            print("No es un triángulo equilátero")
    }
}
```

```

fun main(parametro: Array<String>) {
    val triangulo1 = Triangulo()
    triangulo1.inicializar()
    triangulo1.ladoMayor()
    triangulo1.esEquilatero()
}

```

Ejecución

```

Ingrese lado 1:12
Ingrese lado 2:12
Ingrese lado 3:12
Lado mayor:12
Es un triángulo equilátero

```

Problema propuesto

Implementar una clase llamada Alumno que tenga como propiedades su nombre y su nota. Definir los métodos para inicializar sus propiedades por teclado, imprimirlas y mostrar un mensaje si está regular (nota mayor o igual a 4)

Definir dos objetos de la clase Alumno.

Ejercicio 111 - Clase Alumno

```

// Se define una clase llamada Alumno
class Alumno {

    // Atributos de cada objeto Alumno: su nombre y su nota
    var nombre: String = ""
    var nota: Int = 0

    // Método para cargar los datos del alumno desde teclado
    fun inicializar() {
        print("Ingrese el nombre del alumno: ")
        nombre = readln().toString() // Se lee y guarda el nombre

        print("Ingrese su nota: ")
        nota = readln().toInt() // Se lee y guarda la nota como número entero
    }

    // Método para imprimir los datos del alumno
}

```

```

fun imprimir() {
    println("Alumno: $nombre tiene una nota de $nota")
}

// Método para indicar si el alumno está en estado REGULAR
fun mostrarEstado () {
    // Si la nota es 4 o más, se considera "REGULAR"
    if (nota >= 4)
        println("$nombre se encuentra en estado REGULAR")
    else
        println("$nombre no está REGULAR") // Si no, se considera que no
está regular
}

// Función principal del programa
fun main(parametros: Array<String>) {
    // Se crea un primer objeto Alumno
    val alumno1 = Alumno()
    alumno1.inicializar()      // Se piden los datos al usuario
    alumno1.imprimir()         // Se muestran los datos ingresados
    alumno1.mostrarEstado()    // Se muestra si está o no regular

    // Se repite el mismo proceso para un segundo alumno
    val alumno2 = Alumno()
    alumno2.inicializar()
    alumno2.imprimir()
    alumno2.mostrarEstado()
}

```

Ejecución

```
Ingrese el nombre del alumno: Carlos
Ingrese su nota: 6
Alumno: Carlos tiene una nota de 6
Carlos se encuentra en estado REGULAR
Ingrese el nombre del alumno: Julia
Ingrese su nota: 8
Alumno: Julia tiene una nota de 8
Julia se encuentra en estado REGULAR
```

21. POO - Constructor de la clase

Ejercicio 112 - Clase Persona constructor

```
// Clase Persona con constructor y métodos para imprimir y verificar mayoría de edad

class Persona constructor(nombre: String, edad: Int) {

    var nombre: String = nombre

    var edad: Int = edad

    fun imprimir() {
        println("Nombre: $nombre y tiene una edad de $edad")
    }

    fun esMayorEdad() {
        if (edad >= 18)
            println("Es mayor de edad $nombre")
        else
            println("No es mayor de edad $nombre")
    }
}

fun main(parametro: Array<String>) {
    val personal = Persona("Juan", 12)
    personal.imprimir()
    personal.esMayorEdad()
}
```

Ejecución

```
Nombre: Juan y tiene una edad de 12
No es mayor de edad Juan
```

Ejercicio 113 - Clase Triangulo constructor

```
// Clase Triangulo con constructor, lado mayor y verificación de equilátero
class Triangulo (var lado1: Int, var lado2: Int, var lado3: Int){

    fun ladoMayor() {
        print("Lado mayor:")
        when {
            lado1 > lado2 && lado1 > lado3 -> println(lado1)
            lado2 > lado3 -> println(lado2)
            else -> println(lado3)
        }
    }

    fun esEquilatero() {
        if (lado1 == lado2 && lado1 == lado3)
            print("Es un triángulo equilátero")
        else
            print("No es un triángulo equilátero")
    }
}

fun main(parametro: Array<String>) {
    val triangulo1 = Triangulo(12, 45, 24)
    triangulo1.ladoMayor()
    triangulo1.esEquilatero()
}
```

Ejecución

```
Lado mayor:45
No es un triángulo equilátero
```

Ejercicio 114 - Clase Triangulo primario y secundario

```
// Clase Triangulo con constructor primario y secundario con entrada del
usuario

class Triangulo (var lado1: Int, var lado2: Int, var lado3: Int){

    constructor():this(0, 0, 0) {
        print("Ingrese primer lado:")
        lado1 = readLine()!!.toInt()
    }
}
```

```

print("Ingrese segundo lado:")
lado2 = readLine()!!.toInt()
print("Ingrese tercer lado:")
lado3 = readLine()!!.toInt()
}

fun ladoMayor() {
    print("Lado mayor:")
    when {
        lado1 > lado2 && lado1 > lado3 -> println(lado1)
        lado2 > lado3 -> println(lado2)
        else -> println(lado3)
    }
}

fun esEquilatero() {
    if (lado1 == lado2 && lado1 == lado3)
        println("Es un triángulo equilátero")
    else
        println("No es un triángulo equilátero")
}
}

fun main(parametro: Array<String>) {
    val triangulo1 = Triangulo()
    triangulo1.ladoMayor()
    triangulo1.esEquilatero()
    val triangulo2 = Triangulo(6, 6, 6)
    triangulo2.ladoMayor()
    triangulo2.esEquilatero()
}

```

Ejecución

```

Ingrese primer lado:13
Ingrese segundo lado:12
Ingrese tercer lado:13
Lado mayor:13
No es un triángulo equilátero
Lado mayor:6
Es un triángulo equilátero

```

Problemas propuestos

Implementar una clase llamada Alumno que tenga como propiedades su nombre y su nota. Al constructor llega su nombre y nota.

Imprimir el nombre y su nota. Mostrar un mensaje si está regular (nota mayor o igual a 4)

Definir dos objetos de la clase Alumno.

Cofeccionar una clase que represente un punto en el plano, la coordenada de un punto en el plano está dado por dos valores enteros x e y.

Al constructor llega la ubicación del punto en x e y.

Implementar un método que retorne un String que indique en que cuadrante se ubica dicho punto. (1º Cuadrante si $x > 0 \text{ Y } y > 0$, 2º Cuadrante: $x < 0 \text{ Y } y > 0$, 3º Cuadrante: $x < 0 \text{ Y } y < 0$, 4º Cuadrante: $x > 0 \text{ Y } y < 0$)

Si alguno o los dos valores son cero luego el punto se encuentra en un eje.

Definir luego 5 objetos de la clase Punto en cada uno de los cuadrantes y uno en un eje.

Ejercicio 115 - Clase Alumno y nota

```
// Se define una clase llamada Alumno con dos propiedades que se reciben por el constructor

class Alumno(val nombre: String, val nota: Int) {

    // Método para imprimir el nombre y la nota del alumno
    fun imprimir() {
        println("Alumno: $nombre tiene una nota de $nota")
    }

    // Método para mostrar si el alumno está en estado REGULAR (nota >= 4)
    fun mostrarEstado () {
        if (nota >= 4)
            println("$nombre se encuentra en estado REGULAR")
        else
            println("$nombre no está REGULAR")
    }
}

// Función principal del programa
fun main(parametros: Array<String>) {
    // Se crea un objeto Alumno llamado "alumno1" usando el constructor con datos fijos
    val alumno1 = Alumno("Ana", 7)
```

```

alumno1.imprimir()           // Se imprime su información
alumno1.mostrarEstado()      // Se indica si está REGULAR

// Se crea otro objeto Alumno llamado "alumno2"
val alumno2 = Alumno("Carlos", 2)
alumno2.imprimir()
alumno2.mostrarEstado()
}

```

Ejecución

```

Alumno: Ana tiene una nota de 7
Ana se encuentra en estado REGULAR
Alumno: Carlos tiene una nota de 2
Carlos no está REGULAR

```

Ejercicio 116 - Clase punto coordenada

```

// Se define una clase llamada Punto con dos propiedades: x e y (las
coordenadas)

class Punto(val x: Int, val y: Int) {

    // Esta función determina en qué cuadrante se encuentra el punto (x, y)
    fun retornarCuadrante() = when {
        // Si x > 0 y y > 0, está en el primer cuadrante
        x > 0 && y > 0 -> "Primer cuadrante"

        // Si x < 0 y y > 0, está en el segundo cuadrante
        x < 0 && y > 0 -> "Segundo cuadrante"

        // Si x < 0 y y < 0, está en el tercer cuadrante
        x < 0 && y < 0 -> "Tercer cuadrante"

        // Si x > 0 y y < 0, está en el cuarto cuadrante
        x > 0 && y < 0 -> "Cuarto cuadrante"

        // Si no entra en ningún caso anterior, entonces está sobre un eje
        else -> "Un eje"
    }
}

```

```

    }

}

// Función principal del programa
fun main(parametro: Array<String>) {

    // Se crean 5 objetos de tipo Punto con distintas coordenadas
    val punto1 = Punto(12, 3)      // x > 0, y > 0 → Primer cuadrante
    println("La coordenada ${punto1.x}, ${punto1.y} se encuentra en
    ${punto1.retornarCuadrante() }")

    val punto2 = Punto(-4, 3)      // x < 0, y > 0 → Segundo cuadrante
    println("La coordenada ${punto2.x}, ${punto2.y} se encuentra en
    ${punto2.retornarCuadrante() }")

    val punto3 = Punto(-2, -2)     // x < 0, y < 0 → Tercer cuadrante
    println("La coordenada ${punto3.x}, ${punto3.y} se encuentra en
    ${punto3.retornarCuadrante() }")

    val punto4 = Punto(12, -5)     // x > 0, y < 0 → Cuarto cuadrante
    println("La coordenada ${punto4.x}, ${punto4.y} se encuentra en
    ${punto4.retornarCuadrante() }")

    val punto5 = Punto(0, -5)      // x = 0 → está sobre un eje
    println("La coordenada ${punto5.x}, ${punto5.y} se encuentra en
    ${punto5.retornarCuadrante() }")
}

```

Ejecución

```

La coordenada 12, 3 se encuentra en Primer cuadrante
La coordenada -4, 3 se encuentra en Segundo cuadrante
La coordenada -2, -2 se encuentra en Tercer cuadrante
La coordenada 12, -5 se encuentra en Cuarto cuadrante
La coordenada 0, -5 se encuentra en Un eje

```

22. POO - Llamada de métodos desde otro método de la misma clase

Ejercicio 117 - Clase Operaciones

```
// Clase Operaciones que realiza suma y resta de dos valores
class Operaciones {
    var valor1: Int = 0
    var valor2: Int = 0
    fun cargar() {
        print("Ingrese primer valor:")
        valor1 = readLine()!!.toInt()
        print("Ingrese segundo valor:")
        valor2 = readLine()!!.toInt()
        sumar()
        restar()
    }
    fun sumar() {
        val suma = valor1 + valor2
        println("La suma de $valor1 y $valor2 es $suma")
    }
    fun restar() {
        val resta = valor1 - valor2
        println("La resta de $valor1 y $valor2 es $resta")
    }
}
fun main(parametro: Array<String>) {
    val operaciones1 = Operaciones()
    operaciones1.cargar()
}
```

Ejecución

```
Ingrese primer valor:12
Ingrese segundo valor:22
La suma de 12 y 22 es 34
La resta de 12 y 22 es -10
```

Problema propuesto

Declarar una clase llamada Hijos. Definir dentro de la misma un arreglo para almacenar las edades de 5 personas.

Definir un método cargar donde se ingrese por teclado el arreglo de las edades y llame a otros dos método que impriman la mayor edad y el promedio de edades.

Ejercicio 118 - Clase Hijos

```
// Se define una clase llamada Hijos
class Hijos {
    // Se crea un arreglo de 5 enteros para guardar las edades de los hijos
    val edades = IntArray(5)

    // Función que permite cargar las edades desde la entrada del usuario
    fun cargar() {
        for(i in edades.indices) { // Recorre cada posición del arreglo
            print("Ingrese edad: ")
            edades[i] = readln().toInt() // Lee y guarda la edad ingresada
        }
        // Llama a las funciones que procesan la información
        mayorEdad()
        promedio()
    }

    // Función que encuentra y muestra la mayor edad del arreglo
    fun mayorEdad() {
        var mayor = edades[0] // Se asume inicialmente que la primera edad es
        la mayor
        for(i in edades.indices)
            if (edades[i] > mayor)
                mayor = edades[i] // Se actualiza si se encuentra una edad
        mayor
        println("El hijo con mayor edad tiene $mayor años")
    }

    // Función que calcula y muestra el promedio de las edades
    fun promedio() {
        var suma = 0
        for(i in edades.indices)
            suma += edades[i] // Suma todas las edades
        val promedio = suma / edades.size // Divide por el total de hijos (5)
        println("La edad promedio de los hijos es $promedio")
    }
}
// Función principal del programa
fun main(parametros: Array<String>) {
    // Se crea una instancia de la clase Hijos
    val hijos1 = Hijos()

    // Se llama a la función cargar para ingresar y procesar las edades
    hijos1.cargar()
}
```

Ejecución

```
Ingrese edad: 9
Ingrese edad: 10
Ingrese edad: 14
Ingrese edad: 18
Ingrese edad: 19
El hijo con mayor edad tiene 19 años
La edad promedio de los hijos es 14
```

23. POO - Colaboración de clases

Ejercicio 119 - Clase Cliente

```
// Clase Cliente y Banco con operaciones de depósito y extracción
class Cliente(var nombre: String, var monto: Float) {
    fun depositar(monto: Float) {
        this.monto += monto
    }
    fun extraer(monto: Float) {
        this.monto -= monto
    }
    fun imprimir() {
        println("$nombre tiene depositado la suma de $monto")
    }
}
class Banco {
    val cliente1: Cliente = Cliente("Juan", 0f)
    var cliente2: Cliente = Cliente("Ana", 0f)
    var cliente3: Cliente = Cliente("Luis", 0f)
    fun operar() {
        cliente1.depositar(100f)
        cliente2.depositar(150f)
        cliente3.depositar(200f)
        cliente3.extraer(150f)
    }
    fun depositosTotales() {
        val total = cliente1.monto + cliente2.monto + cliente3.monto
        println("El total de dinero del banco es: $total")
        cliente1.imprimir()
        cliente2.imprimir()
        cliente3.imprimir()
    }
}
fun main(parametro: Array<String>) {
    val bancol = Banco()
    bancol.operar()
```

```
    banco1.depositosTotales()
}
```

Ejecución

```
El total de dinero del banco es: 300.0
Juan tiene depositado la suma de 100.0
Ana tiene depositado la suma de 150.0
Luis tiene depositado la suma de 50.0
```

Ejercicio 120 - Clase Dado

```
// Simulación de un juego de dados

// Clase Dado que puede generar un valor aleatorio entre 1 y 6
class Dado (var valor: Int) {
    fun tirar() {
        valor = ((Math.random() * 6) + 1).toInt()
        imprimir()
    }

    fun imprimir() {
        println("Valor del dado: $valor")
    }
}

// Clase JuegoDeDados que utiliza 3 dados y verifica si todos los valores son
iguales
class JuegoDeDados {
    val dado1 = Dado(1)
    val dado2 = Dado(1)
    val dado3 = Dado(1)

    fun jugar() {
        dado1.tirar()
        dado2.tirar()
        dado3.tirar()

        if (dado1.valor == dado2.valor && dado2.valor == dado3.valor)
            println("Ganó")
        else
            print("Perdió")
    }
}

fun main(parametro: Array<String>) {
    val juegol = JuegoDeDados()
    juegol.jugar()
}
```

Ejecución

```
Valor del dado: 6  
Valor del dado: 4  
Valor del dado: 5  
Perdió
```

Problema propuesto

Plantear una clase Club y otra clase Socio.

La clase Socio debe tener los siguientes propiedades: nombre y la antigüedad en el club (en años).

Al constructor de la clase socio hacer que llegue el nombre y su antigüedad.

La clase Club debe tener como propiedades 3 objetos de la clase Socio.

Definir un método en la clase Club para imprimir el nombre del socio con mayor antigüedad en el club.

Ejercicio 121 - Clase socio

```
// Clase que representa a un socio del club con nombre y antigüedad (años)  
class Socio(val nombre: String, val antiguedad: Int) {  
    // Método para imprimir la información del socio  
    fun imprimir() {  
        println("$nombre tiene una antiguedad de $antiguedad años")  
    }  
}  
  
// Clase que representa un club con tres socios fijos  
class Club {  
    // Se crean tres objetos socio con datos predeterminados  
    val socio1 = Socio("Juan", 22)  
    val socio2 = Socio("Ana", 34)  
    val socio3 = Socio("Carlos", 1)  
  
    // Función que muestra los socios y determina quién tiene la mayor  
    antigüedad  
    fun mayorAntiguedad() {  
        // Imprime los datos de cada socio  
        socio1.imprimir()  
        socio2.imprimir()  
        socio3.imprimir()  
  
        // Imprime el encabezado para el socio con mayor antigüedad  
        print("Socio con mayor antiguedad: ")  
  
        // Evalúa cuál socio tiene la mayor antigüedad usando condiciones  
        when {  
            socio1.antiguedad > socio2.antiguedad && socio1.antiguedad >  
            socio3.antiguedad -> print(socio1.nombre)  
            socio2.antiguedad > socio3.antiguedad -> print(socio2.nombre)  
            else -> print(socio3.nombre)  
        }  
    }  
}
```

```

        }
    }

// Función principal que inicia el programa
fun main(parametro: Array<String>) {
    // Se crea un objeto club con los tres socios
    val club1 = Club()

    // Se llama a la función que muestra el socio con mayor antigüedad
    club1.mayorAntiguedad()
}

```

Ejecución

```

Juan tiene una antiguedad de 22 años
Ana tiene una antiguedad de 34 años
Carlos tiene una antiguedad de 1 años
Socio con mayor antiguedad: Ana

```

24. POO - modificadores de acceso private y public

Ejercicio 122 - Clase operaciones

```

// Clase con encapsulamiento que realiza suma y resta de dos valores
class Operaciones {
    private var valor1: Int = 0
    private var valor2: Int = 0

    fun cargar() {
        print("Ingrese primer valor:")
        valor1 = readLine()!!.toInt()
        print("Ingrese segundo valor:")
        valor2 = readLine()!!.toInt()
        sumar()
        restar()
    }

    private fun sumar() {
        val suma = valor1 + valor2
        println("La suma de $valor1 y $valor2 es $suma")
    }

    private fun restar() {
        val resta = valor1 - valor2
        println("La resta de $valor1 y $valor2 es $resta")
    }
}

```

```

        }
    }

fun main(parametro: Array<String>) {
    val operaciones1 = Operaciones()
    operaciones1.cargar()
}

```

Ejecución

```

Ingrese primer valor:3
Ingrese segundo valor:21
La suma de 3 y 21 es 24
La resta de 3 y 21 es -18

```

Ejercicio 123 - Clase Dado impresión

```

// Clase Dado con método de impresión decorativa
class Dado {
    private var valor: Int = 1

    fun tirar() {
        valor = ((Math.random() * 6) + 1).toInt()
    }

    fun imprimir() {
        separador()
        println("Valor del dado: $valor")
        separador()
    }

    private fun separador() =
        println("*****")
}

fun main(parametro: Array<String>) {
    val dadol = Dado()
    dadol.tirar()
    dadol.imprimir()
}

```

Ejecución

```

*****
Valor del dado: 6
*****

```

Problema propuesto

Desarrollar una clase que defina una propiedad privada de tipo arreglo de 5 enteros. En el bloque init llamar a un método privado que cargue valores aleatorios comprendidos entre 0 y 10.

Definir otros tres métodos públicos que muestren el arreglo, el mayor y el menor elemento.

Ejercicio 124 - Clase vector

```
// Definimos una clase llamada 'Vector'  
class Vector {  
  
    // Se declara un arreglo privado de 5 enteros  
    private val vec = IntArray(5)  
  
    // Bloque de inicialización que se ejecuta automáticamente al crear el  
    objeto  
    init {  
        cargar() // Llama a la función 'cargar' para llenar el arreglo con  
        números aleatorios  
    }  
  
    // Función privada que carga el arreglo con números aleatorios del 0 al 10  
    private fun cargar() {  
        for(i in vec.indices)  
            vec[i] = ((Math.random() * 11)).toInt() // Math.random() genera un  
número entre 0.0 y 1.0, se multiplica por 11 y se convierte a entero  
    }  
  
    // Función para imprimir todos los elementos del arreglo  
    fun imprimir() {  
        println("Listado completo del arreglo")  
        for(i in vec.indices)  
            print("${vec[i]} - ")  
        println() // Salto de linea final  
    }  
  
    // Función para encontrar e imprimir el mayor número del arreglo  
    fun mostrarMayor() {  
        var mayor = vec[0] // Suponemos que el primer elemento es el mayor  
        for(i in vec.indices)  
            if (vec[i] > mayor)  
                mayor = vec[i] // Si encontramos un número mayor, lo  
actualizamos  
        println("El elemento mayor del arreglo es $mayor")  
    }  
  
    // Función para encontrar e imprimir el menor número del arreglo  
    fun mostrarMenor() {  
        var menor = vec[0] // Suponemos que el primer elemento es el menor  
        for(i in vec.indices)
```

```

        if (vec[i] < menor)
            menor = vec[i] // Si encontramos un número menor, lo
actualizamos
        println("El elemento menor del arreglo es $menor")
    }
}

// Función principal donde se ejecuta el programa
fun main(parametro: Array<String>) {
    val vector1 = Vector() // Se crea un objeto de la clase Vector, y se carga
automáticamente
    vector1.imprimir() // Se imprime el contenido del arreglo
    vector1.mostrarMayor() // Se muestra el mayor valor
    vector1.mostrarMenor() // Se muestra el menor valor
}

```

Ejecución

```

Listado completo del arreglo
6 - 7 - 3 - 0 - 1 -
El elemento mayor del arreglo es 7
El elemento menor del arreglo es 0

```

25. POO - propiedades y sus métodos optionales set y get

Ejercicio 125 - Clase Persona personalizada

```

// Clase Persona con propiedades personalizadas usando getter y setter
class Persona {
    var nombre: String = ""
        set(valor) {
            field = valor.toUpperCase()
        }
        get() {
            return "(${field})"
        }

    var edad: Int = 0
        set(valor) {
            if (valor >= 0)
                field = valor
            else
                field = 0
        }
}
```

```

fun main(parametro: Array<String>) {
    val personal = Persona()
    personal.nombre = "juan"
    personal.edad = 23
    println(personal.nombre) // Se imprime: (JUAN)
    println(personal.edad) // Se imprime: 23
    personal.edad = -50
    println(personal.edad) // Se imprime: 0
}

```

Problemas propuestos

Confeccionar una clase que represente un Empleado. Definir como propiedades su nombre y su sueldo.

No permitir que se cargue un valor negativo en su sueldo.

Codificar el método imprimir en la clase.

Plantear una clase llamada Dado. Definir una propiedad llamada valor que permita cargar un valor comprendido entre 1 y 6 si llega un valor que no está comprendido en este rango se debe cargar un 1.

Definir dos métodos, uno que genere un número aleatorio entre 1 y 6 y otro que lo imprima.

Al constructor llega el valor inicial que debe tener el dado (tratar de enviarle el número 7)

Ejercicio 126 - Clase Empleados nombre y sueldo

```

// Se define una clase llamada Empleado que recibe dos parámetros: nombre y
suelo
class Empleado(var nombre: String, sueldo: Double) {

    // Se declara una propiedad 'sueldo' que va a tener lógica personalizada
    para establecer su valor
    var sueldo: Double = 0.0
        set(valor) { // Este es el "setter" del sueldo
            if (valor < 0) // Si se intenta asignar un sueldo negativo
                field = 0.0 // Se asigna 0.0 por defecto (no se permite sueldo
negativo)
            else
                field = valor // Si el sueldo es válido, se asigna normalmente
        }

    // Bloque de inicialización que se ejecuta cuando se crea un objeto de la
    clase
    init {
        this.sueldo = sueldo // Llama al setter para asignar el sueldo
        (verifica si es válido o no)
    }
}

```

```

// Método que imprime el nombre y el sueldo del empleado
fun imprimir() {
    println("$nombre tiene un sueldo de $sueldo")
}
}

// Función principal del programa
fun main(parametro: Array<String>) {
    // Se crea un objeto 'empleado1' con un sueldo válido
    val empleado1 = Empleado("Juan", 12000.50)
    empleado1.imprimir() // Imprime: Juan tiene un sueldo de 12000.5

    // Se crea un objeto 'empleado2' con un sueldo negativo
    val empleado2 = Empleado("Ana", -1200.0)
    empleado2.imprimir() // Imprime: Ana tiene un sueldo de 0.0 (porque el
    setter lo corrigió)
}

```

Ejecución

```

Juan tiene un sueldo de 12000.5
Ana tiene un sueldo de 0.0

```

Ejercicio 127 - Clase Dado valor inicial

```

// Se define una clase llamada Dado que recibe un valor inicial al ser creada
class Dado(valor: Int) {

    // Propiedad 'valor' del dado. Se inicializa con 1 como valor por defecto.
    var valor: Int = 1
        set(valor) { // Setter personalizado para validar el valor del dado
            if (valor in 1..6) // Si el valor está entre 1 y 6 (inclusive)
                field = valor // Se asigna normalmente
            else
                field = 1 // Si no es válido (como 0 o 7), se asigna el valor
por defecto: 1
        }

    // Bloque de inicialización que se ejecuta al crear un objeto de esta
clase
    init {
        this.valor = valor // Se utiliza el setter para validar el valor
pasado al constructor
    }

    // Función que "lanza" el dado, generando un número aleatorio entre 1 y 6
}
```

```

fun tirar() {
    valor = ((Math.random() * 6) + 1).toInt() // Math.random() genera un
número de 0.0 a <1.0
    // Multiplicamos por 6 y sumamos 1 para obtener valores entre 1 y 6
}

// Función que imprime el valor actual del dado
fun imprimir() = println("Valor del dado: $valor")
}

// Función principal
fun main(parametro: Array<String>) {
    val dado1 = Dado(7) // Se intenta crear un dado con valor 7 (inválido),
así que se asigna 1
    dado1.imprimir()      // Muestra: Valor del dado: 1

    dado1.tirar()          // Se lanza el dado (asigna un valor aleatorio entre
1 y 6)
    dado1.imprimir()      // Imprime el nuevo valor obtenido
}

```

Ejecución

```

Valor del dado: 1
Valor del dado: 4

```

26. POO - data class

Ejercicio 128 - Data class articulo

```

// Uso de data class y copia de objetos
data class Articulo(var codigo: Int, var descripcion: String, var precio:
Float)

fun main(parametro: Array<String>) {
    val articulo1 = Articulo(1, "papas", 34f)
    var articulo2 = Articulo(2, "manzanas", 24f)
    println(articulo1)
    println(articulo2)

    val puntero = articulo1

```

```

puntero.precio = 100f
println(articulo1)

var articulo3 = articulo1.copy()
articulo1.precio = 200f
println(articulo1)
println(articulo3)

if (articulo1 == articulo3)
    println("Son iguales $articulo1 y $articulo3")
else
    println("Son distintos $articulo1 y $articulo3")

articulo3.precio = 200f

if (articulo1 == articulo3)
    println("Son iguales $articulo1 y $articulo3")
else
    println("Son distintos $articulo1 y $articulo3")
}

```

Ejecución

```

Articulo(codigo=1, descripcion=papas, precio=34.0)
Articulo(codigo=2, descripcion=manzanas, precio=24.0)
Articulo(codigo=1, descripcion=papas, precio=100.0)
Articulo(codigo=1, descripcion=papas, precio=200.0)|
Articulo(codigo=1, descripcion=papas, precio=100.0)
Son distintos Articulo(codigo=1, descripcion=papas, precio=200.0) y Articulo(codigo=1, descripcion=papas, precio=100.0)
Son iguales Articulo(codigo=1, descripcion=papas, precio=200.0) y Articulo(codigo=1, descripcion=papas, precio=200.0)

```

Ejercicio 129 - Data class persona

```

// Data class Persona con método toString personalizado
data class Persona(var nombre: String, var edad: Int) {
    override fun toString(): String {
        return "$nombre, $edad"
    }
}

fun main(parametro: Array<String>) {
    val personal1 = Persona("Juan", 22)
    val persona2 = Persona("Ana", 59)
    println(personal1)
    println(persona2)
}

```

Ejecución

```

Juan, 22
Ana, 59

```

Problema propuesto

Plantear un data class llamado Dado con una única propiedad llamada valor. Sobreescribir el método `toString` para que muestre tantos asteriscos como indica la propiedad valor.

Ejercicio 130 - Data class Dado valor

```
// Clase 'Dado' como data class, con un solo atributo: 'valor'  
data class Dado (var valor: Int) {  
  
    // Sobrescribe el método toString() para personalizar cómo se imprime el  
    // objeto  
    override fun toString(): String {  
        var cadena = ""  
        for(i in 1..valor)  
            cadena += "*" // Agrega un asterisco por cada número en el valor  
        return cadena  
    }  
}  
fun main(parametro: Array<String>) {  
    val dado1 = Dado(4)    // 4 asteriscos  
    val dado2 = Dado(6)    // 6 asteriscos  
    val dado3 = Dado(1)    // 1 asterisco  
    println(dado1)      // ****  
    println(dado2)      // *****  
    println(dado3)      // *  
}
```

Ejecución

```
****  
*****  
*
```

27. POO - enum class

Ejercicio 131 - Enum class TipoCarta

```
// Uso de enum class para representar tipos de cartas  
enum class TipoCarta {  
    DIAMANTE,  
    TREBOL,  
    CORAZON,  
    PICA  
}
```

```
// Clase Carta que imprime su tipo y valor
class Carta(val tipo: TipoCarta, val valor: Int) {
    fun imprimir() {
        println("Carta: $tipo y su valor es $valor")
    }
}

fun main(parametro: Array<String>) {
    val cartal = Carta(TipoCarta.TREBOL, 4)
    cartal.imprimir()
}
```

Ejecución

```
Carta: TREBOL y su valor es 4
```

Ejercicio 132 - Enum class TipoOperacion

```
// Enum class con operaciones matemáticas y sus símbolos
enum class TipoOperacion(val tipo: String) {
    SUMA("+"),
    RESTA("-"),
    MULTIPLICACION("*"),
    DIVISION("/")
}

// Clase Operacion que realiza la operación seleccionada
class Operacion(val valor1: Int, val valor2: Int, val tipoOperacion: TipoOperacion) {
    fun operar() {
        var resultado: Int = 0
        when (tipoOperacion) {
            TipoOperacion.SUMA -> resultado = valor1 + valor2
            TipoOperacion.RESTA -> resultado = valor1 - valor2
            TipoOperacion.MULTIPLICACION -> resultado = valor1 * valor2
            TipoOperacion.DIVISION -> resultado = valor1 / valor2
        }
        println("$valor1 ${tipoOperacion.tipo} $valor2 es igual a $resultado")
    }
}

fun main(parametro: Array<String>) {
    val operacion1 = Operacion(10, 4, TipoOperacion.SUMA)
    operacion1.operar()
}
```

Ejecución

```
10 + 4 es igual a 14
```

Problema propuesto

Definir un enum class almacenando como constante los nombres de varios países sudamericanos y como propiedad para cada país la cantidad de habitantes que tiene.

Definir una variable de este tipo e imprimir la constante y la cantidad de habitantes de dicha variable.

Ejercicio 133 - Enum class Paises

```
// Se define una enumeración (enum class) llamada "Paises".  
// Cada constante del enum representa un país y tiene asociado un valor  
// entero: su cantidad de habitantes.  
// Este valor se pasa como parámetro al constructor del enum (val habitantes:  
// Int).  
enum class Paises (val habitantes: Int) {  
    BRASIL (202450649),  
    COLOMBIA (50364000),  
    PERU (31151643),  
    VENEZUELA (31028337),  
    CHILE (18261884),  
    ECUADOR (16298217),  
    BOLIVIA (10888000),  
    PARAGUAY (6460000),  
    URUGUAY (3372000)  
}  
  
// Función principal del programa, donde inicia la ejecución.  
fun main(parametro: Array<String>) {  
  
    // Se crea una variable "pais1" y se le asigna el valor BRASIL de la  
    // enumeración Paises.  
    val pais1 = Paises.BRASIL  
  
    // Se imprime el nombre del país (en este caso, "BRASIL").  
    println(pais1)  
  
    // Se imprime la cantidad de habitantes asociados a BRASIL (202450649).  
    println(pais1.habitantes)  
}
```

Ejecución

```
BRASIL  
202450649
```

28. POO - objeto nombrado

Ejercicio 134 - Object Matematica

```
// Objeto singleton con constantes y funciones
object Matematica {
    val PI = 3.1416

    // Genera un número aleatorio entre mínimo y máximo (ambos incluidos)
    fun aleatorio(minimo: Int, maximo: Int) = ((Math.random() * (maximo + 1 - minimo)) + minimo).toInt()

    fun main(parametro: Array<String>) {
        println("El valor de Pi es ${Matematica.PI}")
        print("Un valor aleatorio entre 5 y 10: ")
        println(Matematica.aleatorio(5, 10))
    }
}
```

Ejecución

```
El valor de Pi es 3.1416
Un valor aleatorio entre 5 y 10: 10
```

Ejercicio 135 - Objeto dados

```
// Objeto anónimo que simula tirar 5 dados y determinar el mayor valor
fun main(parametro: Array<String>) {
    val dados = object {
        val arreglo = IntArray(5)

        fun generar() {
            for (i in arreglo.indices)
                arreglo[i] = ((Math.random() * 6) + 1).toInt()
        }

        fun imprimir() {
            for (elemento in arreglo)
                print("$elemento - ")
            println()
        }

        fun mayor(): Int {
            var may = arreglo[0]
            for (i in arreglo.indices)
                if (arreglo[i] > may)
```

```

        may = arreglo[i]
    return may
}

dados.gerar()
dados.imprimir()
print("Mayor valor:")
println(dados.mayor())
}

```

Ejecución

```

1 - 5 - 4 - 1 - 4 -
Mayor valor:5

```

Problema propuesto

Definir un objeto nombrado:

```
object Mayor {
```

con tres métodos llamados "maximo" con dos parámetros cada uno. Los métodos difieren en que uno recibe tipos de datos Int, otro de tipos Float y finalmente el último recibe tipos de datos Double. Los tres métodos deben retornar el mayor de los dos datos que reciben.

Ejercicio 136 - Object Mayor

```

// Se define un objeto "Mayor" usando `object`, lo que significa que es un
singleton (una única instancia).
// Dentro de él hay tres funciones llamadas "maximo", sobrecargadas para
distintos tipos de datos:
// Int, Float y Double. Cada una retorna el valor mayor entre dos números del
mismo tipo.
object Mayor {
    // Función que recibe dos enteros y retorna el mayor.
    fun maximo(x1: Int, x2: Int) = if (x1 > x2) x1 else x2

    // Función que recibe dos números flotantes (Float) y retorna el mayor.
    fun maximo(x1: Float, x2: Float) = if (x1 > x2) x1 else x2

    // Función que recibe dos números decimales (Double) y retorna el mayor.
    fun maximo(x1: Double, x2: Double) = if (x1 > x2) x1 else x2
}

// Función principal del programa donde se hacen pruebas con el objeto
"Mayor".
fun main(parametro: Array<String>) {
    // Llama a la función maximo con dos enteros (Int): 4 y 5.
    // Retorna e imprime el mayor de los dos: 5
}

```

```

    println(Mayor.maximo(4, 5))

    // Llama a la función maximo con dos flotantes (Float): 10.2f y 23.5f.
    // Retorna e imprime el mayor: 23.5
    println(Mayor.maximo(10.2f, 23.5f))

    // Llama a la función maximo con dos dobles (Double): 4.5 y 5.2.
    // Retorna e imprime el mayor: 5.2
    println(Mayor.maximo(4.5, 5.2))
}

```

Ejecución

```

5
23.5
5.2

```

29. POO - herencia

Ejercicio 137 - Clase base Persona

```

// Ejemplo de herencia y sobrescritura de métodos

// Clase base Persona
open class Persona(val nombre: String, val edad: Int) {
    // Método abierto para imprimir los datos
    open fun imprimir() {
        println("Nombre: $nombre")
        println("Edad: $edad")
    }
}

// Clase derivada Empleado que hereda de Persona
class Empleado(nombre: String, edad: Int, val sueldo: Double):
    Persona(nombre, edad) {
    // Sobrescribe el método imprimir para agregar el sueldo
    override fun imprimir() {
        super.imprimir()
        println("Sueldo: $sueldo")
    }

    // Método que determina si el empleado paga impuestos
    fun pagaImpuestos() {
        if (sueldo > 3000)
            println("El empleado $nombre paga impuestos")
        else
            println("El empleado $nombre no paga impuestos")
    }
}

```

```

}

// Función principal
fun main(parametro: Array<String>) {
    val personal = Persona("Jose", 22)
    println("Datos de la persona")
    personal.imprimir()

    val empleado1 = Empleado("Ana", 30, 5000.0)
    println("Datos del empleado")
    empleado1.imprimir()
    empleado1.pagaImpuestos()
}

```

Ejecución

```

Datos de la persona
Nombre: Jose
Edad: 22
Datos del empleado
Nombre: Ana
Edad: 30
Sueldo: 5000.0
El empleado Ana paga impuestos

```

Ejercicio 138 - open class Calculadora

```

// Herencia con una calculadora básica y una científica

// Clase base Calculadora con operaciones básicas
open class Calculadora(val valor1: Double, val valor2: Double) {
    var resultado: Double = 0.0

    fun sumar() { resultado = valor1 + valor2 }
    fun restar() { resultado = valor1 - valor2 }
    fun multiplicar() { resultado = valor1 * valor2 }
    fun dividir() { resultado = valor1 / valor2 }

    fun imprimir() { println("Resultado: $resultado") }
}

// Subclase CalculadoraCientifica con más funciones
class CalculadoraCientifica(valor1: Double, valor2: Double):
    Calculadora(valor1, valor2) {
    fun cuadrado() {
        resultado = valor1 * valor1
    }

    fun raiz() {
        resultado = Math.sqrt(valor1)
    }
}

```

```
// Función principal
fun main(parametro: Array<String>) {
    println("Prueba de la clase Calculadora (suma de dos números)")
    val calculadora1 = Calculadora(10.0, 2.0)
    calculadora1.sumar()
    calculadora1.imprimir()

    println("Prueba de la clase Calculadora Científica")
    val calculadoraCientifica1 = CalculadoraCientifica(10.0, 2.0)
    calculadoraCientifica1.sumar()
    calculadoraCientifica1.imprimir()
    calculadoraCientifica1.cuadrado()
    calculadoraCientifica1.imprimir()
    calculadoraCientifica1.raiz()
    calculadoraCientifica1.imprimir()
}
```

Ejecución

```
Prueba de la clase Calculadora (suma de dos números)
Resultado: 12.0
Prueba de la clase Calculadora Científica
Resultado: 12.0
Resultado: 100.0
Resultado: 3.1622776601683795
```

Problema propuesto

Declarar una clase Dado que genere un valor aleatorio entre 1 y 6, mostrar su valor. Crear una segunda clase llamada DadoRecuadro que genere un valor entre 1 y 6, mostrar el valor recuadrado en asteriscos. Utilizar la herencia entre estas dos clases.

Ejercicio 139 - open class Dado

```
// Clase base abierta llamada Dado.
// La palabra `open` permite que otras clases hereden de ella.
open class Dado {
    // Propiedad protegida, accesible desde esta clase y sus subclases.
    protected var valor: Int = 1

    // Método que genera un número aleatorio entre 1 y 6 para simular una
    // tirada de dado.
    fun tirar() {
        valor = ((Math.random() * 6) + 1).toInt()
    }

    // Método que imprime el valor del dado. Es `open` para que pueda ser
    // sobreescrito en subclases.
```

```

open fun imprimir() {
    println("$valor")
}
}

// Subclase que hereda de Dado.
// Sobrescribe el método `imprimir` para mostrar el valor con un marco o
// recuadro.
class DadoRecuadro: Dado() {
    // Sobrescribe el método imprimir para mostrar el valor entre líneas de
    // asteriscos.
    override fun imprimir() {
        println("****")
        println("*$valor*") // Accede a `valor` porque es `protected`.
        println("****")
    }
}

// Función principal donde se crean e imprimen dos dados.
fun main(parametro: Array<String>) {
    val dado1 = Dado()          // Se crea un objeto de la clase base Dado
    dado1.tirar()               // Se genera un valor aleatorio
    dado1.imprimir()           // Se imprime solo el valor, por ejemplo: 3

    val dado2 = DadoRecuadro()  // Se crea un objeto de la subclase
DadoRecuadro
    dado2.tirar()              // Se genera un valor aleatorio
    dado2.imprimir()           // Se imprime el valor en forma de recuadro:
    // ***
    // *4*
    // ***
}

```

Ejecución

```

2
****
*6*
****

```

30. POO - herencia - clases abstractas

Ejercicio 140 - open class Dado

```

// Ejemplo de clase abstracta y herencia para operaciones matemáticas

// Clase abstracta que define una operación
abstract class Operacion(val valor1: Int, val valor2: Int) {
    protected var resultado: Int = 0

    abstract fun operar()

    fun imprimir() {
        println("Resultado: $resultado")
    }
}

```

```

        }

// Clase Suma que implementa la operación
class Suma(valor1: Int, valor2: Int): Operacion(valor1, valor2) {
    override fun operar() {
        resultado = valor1 + valor2
    }
}

// Clase Resta que implementa la operación
class Resta(valor1: Int, valor2: Int): Operacion(valor1, valor2) {
    override fun operar() {
        resultado = valor1 - valor2
    }
}

// Función principal
fun main(parametro: Array<String>) {
    val sumal = Suma(10, 4)
    sumal.operar()
    sumal.imprimir()

    val restal = Resta(20, 5)
    restal.operar()
    restal.imprimir()
}

```

Ejecución

```

Resultado: 14
Resultado: 15

```

Problema propuesto

Declarar una clase abstracta Cuenta y dos subclases CajaAhorra y PlazoFijo. Definir las propiedades y métodos comunes entre una caja de ahorro y un plazo fijo y agruparlos en la clase Cuenta.

Una caja de ahorro y un plazo fijo tienen un nombre de titular y un monto. Un plazo fijo añade un plazo de imposición en días y una tasa de interés. Hacer que la caja de ahorro no genera intereses.

En la función main del programa definir un objeto de la clase CajaAhorro y otro de la clase PlazoFijo.

Ejercicio 141 - abstract class Cuenta

```

// Clase abstracta base llamada Cuenta
// No se puede instanciar directamente, solo a través de sus subclases.
abstract class Cuenta(val titular: String, val monto: Double) {

```

```

// Método que puede ser sobreescrito en las subclases.
open fun imprimir() {
    println("Titular: $titular")
    println("Monto: $monto")
}
// Subclase llamada CajaAhorro que hereda de Cuenta.
class CajaAhorro(titular: String, monto: Double): Cuenta(titular, monto) {

    // Sobrescribe el método imprimir para mostrar un mensaje adicional.
    override fun imprimir() {
        println("Cuenta de caja de ahorro")
        super.imprimir() // Llama al método imprimir de la clase base
    }
}
// Subclase llamada PlazoFijo que también hereda de Cuenta.
// Además, tiene atributos adicionales: plazo (en días) e interes (porcentaje).
class PlazoFijo(titular: String, monto: Double, val plazo: Int, val interes: Double): Cuenta(titular, monto) {

    // Sobrescribe el método imprimir para mostrar más detalles específicos.
    override fun imprimir() {
        println("Cuenta de plazo fijo")
        println("Plazo en días: $plazo")
        println("Intereses: $interes")

        // Calcula la ganancia en base al monto e interés.
        val ganancia = monto * interes / 100
        println("Importe del interés: $ganancia")

        super.imprimir() // Llama a la función imprimir de la clase padre
    }
}
// Función principal donde se crean e imprimen las cuentas
fun main(parametro: Array<String>) {

    // Se crea un objeto de tipo CajaAhorro
    val cajaAhorro1 = CajaAhorro("juan", 10000.0)
    cajaAhorro1.imprimir()

    // Se crea un objeto de tipo PlazoFijo
    val plazoFijo1 = PlazoFijo("Ana", 5000.0, 30, 1.23)
    plazoFijo1.imprimir()
}

```

Ejecución

```
Cuenta de caja de ahorro
Titular: juan
Monto: 10000.0
Cuenta de plazo fijo
Plazo en dias: 30
Intereses: 1.23
Importe del interés: 61.5
Titular: Ana
Monto: 5000.0
```

31. POO - declaración e implementación de interfaces

Ejercicio 142 - interface Punto

```
// Interfaces con diferentes implementaciones de impresión

// Interfaz Punto con método imprimir
interface Punto {
    fun imprimir()
}

// Implementación para un punto en el plano 2D
class PuntoPlano(val x: Int, val y: Int): Punto {
    override fun imprimir() {
        println("Punto en el plano: ($x, $y)")
    }
}

// Implementación para un punto en el espacio 3D
class PuntoEspacio(val x: Int, val y: Int, val z: Int): Punto {
    override fun imprimir() {
        println("Punto en el espacio: ($x, $y, $z)")
    }
}

fun main(parametro: Array<String>) {
    val puntoPlano1 = PuntoPlano(10, 4)
    puntoPlano1.imprimir()
    val puntoEspacio1 = PuntoEspacio(20, 50, 60)
    puntoEspacio1.imprimir()
}
```

Ejecución

```
Punto en el plano: (10,4)
Punto en el espacio: (20,50,60)
```

Ejercicio 143 - interface Figura

```
// Uso de interfaz Figura con clases que implementan superficie y perímetro

// Interfaz que define las operaciones comunes a las figuras geométricas
interface Figura {
    fun calcularSuperficie(): Int
    fun calcularperimetro(): Int
    fun tituloResultado() {
        println("Datos de la figura")
    }
}

// Implementación para un cuadrado
class Cuadrado(val lado: Int): Figura {
    override fun calcularSuperficie(): Int {
        return lado * lado
    }
    override fun calcularperimetro(): Int {
        return lado * 4
    }
}

// Implementación para un rectángulo
class Rectangulo(val ladoMayor: Int, val ladoMenor: Int): Figura {
    override fun calcularSuperficie(): Int {
        return ladoMayor * ladoMenor
    }
    override fun calcularperimetro(): Int {
        return (ladoMayor * 2) + (ladoMenor * 2)
    }
}

fun main(parametro: Array<String>) {
    val cuadrado1 = Cuadrado(10)
    cuadrado1.tituloResultado()
    println("Perímetro del cuadrado : ${cuadrado1.calcularperimetro()}")
    println("Superficie del cuadrado : ${cuadrado1.calcularSuperficie()}")

    val rectangulo1 = Rectangulo(10, 5)
    rectangulo1.tituloResultado()
    println("Perímetro del rectángulo : ${rectangulo1.calcularperimetro()}")
    println("Superficie del rectángulo : ${rectangulo1.calcularSuperficie()}")
}
```

Ejecución

```
Datos de la figura
Perímetro del cuadrado : 40
Superficie del cuadrado : 100
Datos de la figura
Perímetro del rectángulo : 30
Superficie del rectángulo : 50
```

Ejercicio 144 - Mayores de edad

```
// Arreglos de objetos con lógica para contar mayores de edad

// Clase Persona con método para verificar si es mayor de edad
class Persona(val nombre: String, val edad: Int) {
    fun imprimir() {
        println("Nombre: $nombre Edad: $edad")
    }

    fun esMayor() = edad >= 18
}

fun main(parametro: Array<String>) {
    val personas: Array<Persona> = arrayOf(
        Persona("ana", 22),
        Persona("juan", 13),
        Persona("carlos", 6),
        Persona("maria", 72)
    )

    println("Listado de personas")
    for (per in personas)
        per.imprimir()

    var cant = 0
    for (per in personas)
        if (per.esMayor())
            cant++

    println("Cantidad de personas mayores de edad: $cant")
}
```

Ejecución

```
Listado de personas
Nombre: ana Edad: 22
Nombre: juan Edad: 13
Nombre: carlos Edad: 6
Nombre: maria Edad: 72
Cantidad de personas mayores de edad: 2
```

Problemas propuestos

Se tiene la declaración del siguiente data class:

```
data class Articulo(val codigo: Int, val descripcion: String, var precio: Float)
```

Definir un Array con 4 elementos de tipo Articulo.

Implementar dos funciones, una que le envíemos el Array y nos muestre todos sus componentes, y otra que también reciba el Array de artículos y proceda a aumentar en 10% a todos los productos.

Declarar una clase Dado que tenga como propiedad su valor y dos métodos que permitan tirar el dado e imprimir su valor.

En la main definir un Array con 5 objetos de tipo Dado.

Tirar los 5 dados e imprimir los valores de cada uno.

Ejercicio 145 - data class Articulo

```
// Se define una clase de datos llamada Articulo.  
// Cada artículo tiene:  
// - un código (entero),  
// - una descripción (texto),  
// - y un precio (decimal tipo float).  
data class Articulo(val codigo: Int, val descripcion: String, var precio: Float)  
  
// Función que recibe un arreglo de artículos y los imprime uno por uno.  
fun imprimir(articulos: Array<Articulo>) {  
    for(articulo in articulos)  
        println("Código: ${articulo.codigo} - Descripción  
${articulo.descripcion} Precio: ${articulo.precio}")  
}  
  
// Esta función también recibe un arreglo de artículos.  
// Recorre cada uno y aumenta su precio un 10%.  
fun aumentarPrecio(articulos: Array<Articulo>) {  
    for(articulo in articulos)  
        articulo.precio = articulo.precio + (articulo.precio * 0.10f)  
}  
  
fun main(parametro: Array<String>) {  
    // Se crea un arreglo de artículos con sus datos iniciales.  
    val articulos: Array<Articulo> = arrayOf(  
        Articulo(1, "papas", 7.5f),  
        Articulo(2, "manzanas", 23.2f),  
        Articulo(1, "naranjas", 12f),  
        Articulo(1, "cebolla", 21f)  
    )  
  
    // Se imprime el listado actual de precios  
    println("Listado de precios actual")  
    imprimir(articulos)  
  
    // Se aumenta el precio en 10% para cada artículo  
    aumentarPrecio(articulos)  
  
    println()
```

```

    println("Listado de precios con aumento del 10%")
        imprimir(articulos) // Se imprime el nuevo listado con precios
actualizados
}

```

Ejecución

```

Listado de precios actual
Código: 1 - Descripción papas Precio: 7.5
Código: 2 - Descripción manzanas Precio: 23.2
Código: 1 - Descripción naranjas Precio: 12.0
Código: 1 - Descripción cebolla Precio: 21.0

Listado de precios con aumento del 10%
Código: 1 - Descripción papas Precio: 8.25
Código: 2 - Descripción manzanas Precio: 25.52
Código: 1 - Descripción naranjas Precio: 13.2
Código: 1 - Descripción cebolla Precio: 23.1

```

Ejercicio 146 - class Dado mutable

```

// Se define una clase llamada Dado con una propiedad mutable llamada
'valor'.
// Por defecto, el valor del dado es 1 (valor inicial si no se pasa ningún
argumento).
class Dado (var valor: Int = 1){

    // Método para simular tirar el dado.
    // Usa Math.random() que genera un número aleatorio entre 0 y 1,
    // lo multiplica por 6 y le suma 1 para obtener un valor entre 1 y 6.
    // Luego convierte ese valor a entero (toInt()).
    fun tirar() {
        valor = ((Math.random() * 6) + 1).toInt()
    }

    // Método para imprimir el valor actual del dado en consola.
    fun imprimir() {
        println("Valor del dado: $valor")
    }
}

fun main(parametro: Array<String>) {
    // Se crea un arreglo de 5 objetos de tipo Dado.
    // Todos los dados se crean con el valor por defecto (1).
    var dados: Array<Dado> = arrayOf(Dado(), Dado(), Dado(), Dado(), Dado())

    // Se recorre el arreglo y se tira cada dado (se le asigna un nuevo valor
aleatorio entre 1 y 6).
    for(dado in dados)
        dado.tirar()

    // Se recorre nuevamente el arreglo y se imprime el valor de cada dado.
}

```

```
    for(dado in dados)
        dado.imprimir()
}
```

Ejecución

```
Valor del dado: 5
Valor del dado: 2
Valor del dado: 1
Valor del dado: 2
Valor del dado: 1
```

32. Funciones de orden superior

Ejercicio 147 - función operar

```
// Paso de funciones como parámetro para realizar operaciones

// Función de orden superior que recibe dos valores y una función
fun operar(v1: Int, v2: Int, fn: (Int, Int) -> Int) : Int {
    return fn(v1, v2)
}

// Funciones matemáticas básicas
fun sumar(x1: Int, x2: Int) = x1 + x2
fun restar(x1: Int, x2: Int) = x1 - x2
fun multiplicar(x1: Int, x2: Int) = x1 * x2
fun dividir(x1: Int, x2: Int) = x1 / x2

fun main(parametro: Array<String>) {
    val resul = operar(10, 5, ::sumar)
    println("La suma de 10 y 5 es $resul")

    val resu2 = operar(5, 2, ::sumar)
    println("La suma de 5 y 2 es $resu2")

    println("La resta de 100 y 40 es ${operar(100, 40, ::restar)}")
    println("El producto entre 5 y 20 es ${operar(5, 20, ::multiplicar)}")
    println("La división entre 10 y 5 es ${operar(10, 5, ::dividir)}")
}
```

Ejecución

```
La suma de 10 y 5 es 15
La suma de 5 y 2 es 7
La resta de 100 y 40 es 60
El producto entre 5 y 20 es 100
La división entre 10 y 5 es 2
```

Ejercicio 148 - class Persona nombre y edad

```
// Se define una clase llamada Persona con dos propiedades: nombre (String) y
edad (Int).
class Persona(val nombre: String, val edad: Int) {

    // Método 'esMayor' que recibe como parámetro una función (lambda o
referencia) que toma un Int y devuelve un Boolean.
    // La función se aplica a la edad de la persona.
    fun esMayor(fn: (Int) -> Boolean): Boolean {
        return fn(edad) // Se evalúa la función pasando la edad de la persona
    }
}

// Función que determina si alguien es mayor en Estados Unidos (21 años o
más)
fun mayorEstadosUnidos(edad: Int): Boolean {
    if (edad >= 21)
        return true
    else
        return false
}

// Función que determina si alguien es mayor en Argentina (18 años o más)
fun mayorArgentina(edad: Int): Boolean {
    if (edad >= 18)
        return true
    else
        return false
}

fun main(parametro: Array<String>) {
    // Se crea una instancia de Persona con nombre "juan" y edad 18
    val personal = Persona("juan", 18)

    // Se evalúa si la persona es mayor en Argentina, usando la función como
parámetro
    if (personal.esMayor(::mayorArgentina))
        println("${personal.nombre} es mayor si vive en Argentina")
    else
        println("${personal.nombre} no es mayor si vive en Argentina")

    // Se evalúa si la persona es mayor en Estados Unidos
    if (personal.esMayor(::mayorEstadosUnidos))
        println("${personal.nombre} es mayor si vive en Estados Unidos")
    else
```

```
        println("${personal.nombre} no es mayor si vive en Estados Unidos")
    }
```

Ejecución

```
juan es mayor si vive en Argentina
juan no es mayor si vive en Estados Unidos
```

33. Expresiones lambda

Ejercicio 149 - class Persona mayoría de edad

```
// Paso de funciones como criterio para determinar mayoría de edad

// Clase Persona con función que recibe un criterio para evaluar edad
class Persona(val nombre: String, val edad: Int) {
    fun esMayor(fn:(Int) -> Boolean): Boolean {
        return fn(edad)
    }
}

// Función que define mayoría de edad en Estados Unidos (21+)
fun mayorEstadosUnidos(edad: Int): Boolean {
    return edad >= 21
}

// Función que define mayoría de edad en Argentina (18+)
fun mayorArgentina(edad: Int): Boolean {
    return edad >= 18
}

fun main(parametro: Array<String>) {
    val personal = Persona("juan", 18)

    if (personal.esMayor(::mayorArgentina))
        println("${personal.nombre} es mayor si vive en Argentina")
    else
        println("${personal.nombre} no es mayor si vive en Argentina")

    if (personal.esMayor(::mayorEstadosUnidos))
        println("${personal.nombre} es mayor si vive en Estados Unidos")
    else
        println("${personal.nombre} no es mayor si vive en Estados Unidos")
}
```

Ejecución

```
juan es mayor si vive en Argentina  
juan no es mayor si vive en Estados Unidos
```

Ejercicio 150 - función imprimirSi

```
// Función de orden superior para imprimir elementos de un arreglo que cumplen una condición  
fun imprimirSi(arreglo: IntArray, fn:(Int) -> Boolean) {  
    for (elemento in arreglo)  
        if (fn(elemento))  
            print("$elemento ")  
    println()  
}  
  
fun main(parametro: Array<String>) {  
    val arreglo1 = IntArray(10)  
    for (i in arreglo1.indices)  
        arreglo1[i] = ((Math.random() * 100)).toInt()  
  
    println("Imprimir los valores múltiplos de 2")  
    imprimirSi(arreglo1) { x -> x % 2 == 0 }  
  
    println("Imprimir los valores múltiplos de 3 o de 5")  
    imprimirSi(arreglo1) { x -> x % 3 == 0 || x % 5 == 0 }  
  
    println("Imprimir los valores mayores o iguales a 50")  
    imprimirSi(arreglo1) { x -> x >= 50 }  
  
    println("Imprimir los valores comprendidos entre 1 y 10, 20 y 30, 90 y 95")  
    imprimirSi(arreglo1) { x ->  
        when (x) {  
            in 1..10, in 20..30, in 90..95 -> true  
            else -> false  
        }  
    }  
  
    println("Imprimir todos los valores")  
    imprimirSi(arreglo1) { true }  
}
```

Ejecución

```
Imprimir los valores múltiplos de 2
70 70 70 14
Imprimir los valores múltiplos de 3 o de 5
70 99 70 70 57 39
Imprimir los valores mayores o iguales a 50
70 99 70 70 57 79 53
Imprimir los valores comprendidos entre 1 y 10, 20 y 30, 90 y 95
29
Imprimir todos los valores
70 99 70 29 70 57 14 39 79 53
```

Ejercicio 151 - función filtrar

```
// Función que filtra caracteres de una cadena según un criterio
fun filtrar(cadena: String, fn: (Char) -> Boolean): String {
    val cad = StringBuilder()
    for (ele in cadena)
        if (fn(ele))
            cad.append(ele)
    return cad.toString()
}

fun main(parametro: Array<String>) {
    val cadena = "¿Esto es la prueba 1 o la prueba 2?"
    println("String original")
    println(cadena)

    val resultado1 = filtrar(cadena) {
        it.lowercaseChar() in "aeiou"
    }
    println("Solo las vocales")
    println(resultado1)

    val resultado2 = filtrar(cadena) {
        it in 'a'..'z'
    }
    println("Solo los caracteres en minúsculas")
    println(resultado2)

    val resultado3 = filtrar(cadena) {
        it !in 'a'..'z' && it !in 'A'..'Z'
    }
    println("Solo los caracteres no alfabéticos")
    println(resultado3)
}
```

Ejecución

```
String original
¿Esto es la prueba 1 o la prueba 2?
Solo las vocales
Eoeaueaoauea
Solo los caracteres en minúsculas
stoeslapruebaolaprueba
Solo los caracteres no alfabéticos
¿    1    2?
```

34. Expresiones lambda con arreglos IntArray, FloatArray, DoubleArray etc.

Ejercicio 152 - Elementos de un arreglo

```
// Evaluación de condiciones sobre elementos de un arreglo

fun main(parametro: Array<String>) {
    val arreglo = IntArray(20) { (Math.random() * 11).toInt() }
    println("Listado completo del arreglo")
    for (elemento in arreglo)
        print("$elemento ")
    println()

    val cant1 = arreglo.count { it <= 5 }
    println("Cantidad de elementos menores o iguales a 5: $cant1")

    if (arreglo.all { it <= 9 })
        println("Todos los elementos son menores o iguales a 9")
    else
        println("No todos los elementos son menores o iguales a 9")

    if (arreglo.any { it == 10 })
        println("Al menos uno de los elementos es un 10")
    else
        println("Todos los elementos son distintos a 10")
}
```

Ejecución

```
Listado completo del arreglo
10 9 7 6 8 7 9 0 0 9 6 0 8 5 0 1 5 0 4 4
Cantidad de elementos menores o iguales a 5: 10
No todos los elementos son menores o iguales a 9
Al menos uno de los elementos es un 10
```

Problema propuesto

Crear un arreglo de tipo `FloatArray` de 10 elementos, cargar sus elementos por teclado.

Imprimir la cantidad de valores ingresados menores a 50.

Imprimir un mensaje si todos los valores son menores a 50.

Ejercicio 153 - función main arreglo de diez

```
fun main(parametro: Array<String>) {
    // Se declara un arreglo de 10 elementos tipo Float (números decimales)
    val arreglo = FloatArray(10)

    // Se usa un bucle for para recorrer los índices del arreglo (0 hasta 9)
    for (i in arreglo.indices) {
        print("Ingrese elemento: ") // Pide al usuario un número
        arreglo[i] = readln().toFloat() // Lee el valor, lo convierte a Float
    }

    // Se imprime el listado completo del arreglo
    println("Listado completo del arreglo")
    for (elemento in arreglo)
        print("$elemento ") // Imprime cada valor separado por un espacio

    println() // Salto de línea para separar las salidas

    // Se cuenta cuántos elementos del arreglo son menores a 50
    val cant = arreglo.count { it < 50 }

    // Se imprime cuántos valores menores a 50 hay
    println("Cantidad de valores ingresados menores a 50: $cant")

    // Se verifica si *todos* los valores del arreglo son menores a 50
    if (arreglo.all { it < 50 })
        println("Todos los valores son menores a 50")
    else
        println("No todos los valores son menores a 50")
}
```

Ejecución

```
Ingrese elemento: 4
Ingrese elemento: 5
Ingrese elemento: 10
Ingrese elemento: 99
Ingrese elemento: 20
Ingrese elemento: 24
Ingrese elemento: 13
Ingrese elemento: 9
Ingrese elemento: 6
Ingrese elemento: 10
Listado completo del arreglo
4.0 5.0 10.0 99.0 20.0 24.0 13.0 9.0 6.0 10.0
Cantidad de valores ingresados menores a 50: 9
No todos los valores son menores a 50
```

35. Expresiones lambda con arreglos IntArray, FloatArray, DoubleArray etc.

Ejercicio 154 - recorrer y procesar un arreglo

```
/ Uso de función de orden superior para recorrer y procesar un arreglo

fun recorrerTodo(arreglo: IntArray, fn: (Int) -> Unit) {
    for (elemento in arreglo)
        fn(elemento)
}

fun main(parametro: Array<String>) {
    val arreglo1 = IntArray(10)
    for (i in arreglo1.indices)
        arreglo1[i] = ((Math.random() * 100)).toInt()

    println("Impresion de todo el arreglo")
    for (elemento in arreglo1)
        print("$elemento ")
    println()

    var cantidad = 0
    recorrerTodo(arreglo1) {
        if (it % 3 == 0)
            cantidad++
    }
    println("La cantidad de elementos múltiplos de 3 son $cantidad")

    var suma = 0
    recorrerTodo(arreglo1) {
```

```

        if (it > 50)
            suma += it
    }
    println("La suma de todos los elementos mayores a 50 es $suma")
}

```

Ejecución

```

Impresion de todo el arreglo
62 32 73 96 33 74 49 47 27 10
La cantidad de elementos múltiplos de 3 son 3
La suma de todos los elementos mayores a 50 es 305

```

Ejercicio 155 - forEach

```

// Uso de forEach para recorrer un arreglo y aplicar condiciones
fun main(parametro: Array<String>) {
    val arreglo1 = IntArray(10)
    for (i in arreglo1.indices)
        arreglo1[i] = ((Math.random() * 100)).toInt()

    println("Impresión de todo el arreglo")
    for (elemento in arreglo1)
        print("$elemento ")
    println()

    var cantidad = 0
    arreglo1.forEach {
        if (it % 3 == 0)
            cantidad++
    }
    println("La cantidad de elementos múltiplos de 3 es $cantidad")

    var suma = 0
    arreglo1.forEach {
        if (it > 50)
            suma += it
    }
    println("La suma de todos los elementos mayores a 50 es $suma")
}

```

Ejercicio 156 - Arreglo de objetos Persona

```

// Arreglo de objetos Persona y uso de forEach para contar mayores de edad
class Persona(val nombre: String, val edad: Int) {
    fun imprimir() {
        println("Nombre: $nombre Edad: $edad")
    }

    fun esMayor() = edad >= 18
}

fun main(parametro: Array<String>) {
    val personas: Array<Persona> = arrayOf(
        Persona("ana", 22),

```

```

        Persona("juan", 13),
        Persona("carlos", 6),
        Persona("maria", 72)
    )
    println("Listado de personas")
    for (per in personas)
        per.imprimir()

    var cant = 0
    personas.forEach {
        if (it.esMayor())
            cant++
    }
    println("Cantidad de personas mayores de edad: $cant")
}

```

Ejecución

```

Listado de personas
Nombre: ana Edad: 22
Nombre: juan Edad: 13
Nombre: carlos Edad: 6
Nombre: maria Edad: 72
Cantidad de personas mayores de edad: 2

```

Problema propuesto

Declarar una clase Dado que tenga como propiedad su valor y dos métodos que permitan tirar el dado e imprimir su valor.

En la main definir un Array con 5 objetos de tipo Dado.

Tirar los 5 dados e imprimir cuantos 1, 2, 3, 4, 5 y 6 salieron.

Ejercicio 157 - Clase Dado valor y dos métodos

```

// Se define la clase Dado
class Dado (var valor: Int = 1) {

    // Método para tirar el dado, genera un valor aleatorio entre 1 y 6
    fun tirar() {
        valor = ((Math.random() * 6) + 1).toInt() // Math.random() devuelve
un número entre 0.0 y 1.0
    }

    // Método para imprimir el valor del dado
    fun imprimir() {
        println("Valor del dado: $valor")
    }
}

```

```

fun main(parametro: Array<String>) {
    // Se crea un arreglo con 5 objetos Dado
    var dados: Array<Dado> = arrayOf(Dado(), Dado(), Dado(), Dado(), Dado())

    // Se tiran los 5 dados (se les asigna un valor aleatorio del 1 al 6)
    for(dado in dados)
        dado.tirar()

    // Se imprime el valor de cada dado
    for(dado in dados)
        dado.imprimir()

    // Variables contadoras para cada posible resultado del dado
    var cant1 = 0
    var cant2 = 0
    var cant3 = 0
    var cant4 = 0
    var cant5 = 0
    var cant6 = 0

    // Se recorre cada dado y se incrementa el contador correspondiente según
    su valor
    dados.forEach {
        when (it.valor) {
            1 -> cant1++
            2 -> cant2++
            3 -> cant3++
            4 -> cant4++
            5 -> cant5++
            6 -> cant6++
        }
    }

    // Se imprime cuántas veces salió cada valor del dado
    println("Cantidad de dados que tienen el valor 1: $cant1")
    println("Cantidad de dados que tienen el valor 2: $cant2")
    println("Cantidad de dados que tienen el valor 3: $cant3")
    println("Cantidad de dados que tienen el valor 4: $cant4")
    println("Cantidad de dados que tienen el valor 5: $cant5")
    println("Cantidad de dados que tienen el valor 6: $cant6")
}

```

Ejecución

```
Valor del dado: 6
Valor del dado: 6
Valor del dado: 2
Valor del dado: 3
Valor del dado: 1
Cantidad de datos que tienen el valor 1: 1
Cantidad de datos que tienen el valor 2: 1
Cantidad de datos que tienen el valor 3: 1
Cantidad de datos que tienen el valor 4: 0
Cantidad de datos que tienen el valor 5: 0
Cantidad de datos que tienen el valor 6: 2
```

36. Funciones de extensión

Ejercicio 158 - Extensión de String

```
// Extensión de String para obtener mitades de una cadena
fun String.mitadPrimera(): String {
    return this.substring(0 until this.length / 2)
}

fun String.segundaMitad(): String {
    return this.substring(this.length / 2 until this.length)
}

fun main(args: Array<String>) {
    val cadena1 = "Viento"
    println(cadena1.mitadPrimera())
    println(cadena1.segundaMitad())
}
```

Ejecución

```
Vie
nto
```

Ejercicio 159 - Extensión de IntArray

```
// Extensión de IntArray para imprimir el contenido del arreglo
fun IntArray.imprimir() {
    print("[")
    for (elemento in this) {
        print("$elemento ")
    }
    println("]")
}
```

```
fun main(args: Array<String>) {
    val arreglo1 = IntArray(10) { it }
    arreglo1.imprimir()
}
```

Ejecución

```
[0 1 2 3 4 5 6 7 8 9 ]
```

Problemas propuestos

Agregar a la clase String un método imprimir que muestre todos los caracteres que tiene almacenado en una línea.

Codificar la función de extensión llamada "hasta" que debe extender la clase Int y tiene por objetivo mostrar desde el valor entero que almacena el objeto hasta el valor que llega como parámetro:

```
fun Int.hasta(fin: Int)
```

Ejercicio 160 - función de extensión

```
// Se define una función de extensión para la clase String
fun String.imprimir() {
    // Dentro de la función, se imprime el valor de la cadena (this hace referencia a la propia cadena)
    println(this)
}

fun main(args: Array<String>) {
    // Se llama a la función imprimir directamente sobre el string "Hola Mundo"
    "Hola Mundo".imprimir()

    // Se crea una variable cadena1 que contiene la cadena "Fin"
    val cadena1 = "Fin"

    // Se llama a la función imprimir sobre la variable cadena1
    cadena1.imprimir()
}
```

Ejecución

```
Hola Mundo
Fin
```

Ejercicio 161 - función Int.hasta

```
// Definimos una función de extensión para la clase Int llamada `hasta`
// Esta función recibe otro entero llamado `fin` como parámetro
```

```

fun Int.hasta(fin: Int) {
    // Recorremos los valores desde el número actual (this) hasta el valor
    'fin'
    for (valor in this..fin)
        print("$valor-") // Imprime cada valor seguido de un guion
    println() // Salto de línea al terminar
}
fun main(args: Array<String>) {
    val v = 10          // Se declara una variable entera con valor 10
    v.hasta(100)       // Se llama a la función `hasta` desde 10 hasta 100
    5.hasta(10)        // También se puede llamar directamente con un número
literal (del 5 al 10)
}

```

Ejecución

10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-

-37-38-39-40-41-42-43-44-45-46-47-48-49-50-51-52-53-54-55-56-57-58-59-60-61-62-63-64-

-65-66-67-68-69-70-71-72-73-74-75-76-77-78-79-80-81-82-83-84-85-86-87-88-89-90-91-92-93-94-95-96-97-98-99-100-

5-6-7-8-9-10-

37. Sobrecarga de operadores

Ejercicio 162 - clase Vector con operadores

```

// Clase Vector con operadores sobrecargados para suma, resta, multiplicación
y división entre vectores
class Vector {
    val arreglo = IntArray(5)

    fun cargar() {
        for (i in arreglo.indices)
            arreglo[i] = (Math.random() * 11 + 1).toInt()
    }

    fun imprimir() {
        for (elemento in arreglo)
            print("$elemento ")
        println()
    }

    operator fun plus(vector2: Vector): Vector {
        val suma = Vector()
        for (i in arreglo.indices)
            suma.arreglo[i] = arreglo[i] + vector2.arreglo[i]
        return suma
    }
}

```

```

operator fun minus(vector2: Vector): Vector {
    val resta = Vector()
    for (i in arreglo.indices)
        resta.arreglo[i] = arreglo[i] - vector2.arreglo[i]
    return resta
}

operator fun times(vector2: Vector): Vector {
    val producto = Vector()
    for (i in arreglo.indices)
        producto.arreglo[i] = arreglo[i] * vector2.arreglo[i]
    return producto
}

operator fun div(vector2: Vector): Vector {
    val division = Vector()
    for (i in arreglo.indices)
        division.arreglo[i] = arreglo[i] / vector2.arreglo[i]
    return division
}

fun main(args: Array<String>) {
    val vec1 = Vector()
    vec1.cargar()
    val vec2 = Vector()
    vec2.cargar()

    vec1.imprimir()
    vec2.imprimir()

    val vecSuma = vec1 + vec2
    println("Suma componente a componente de los dos vectores")
    vecSuma.imprimir()

    val vecResta = vec1 - vec2
    println("La resta componente a componente de los dos vectores")
    vecResta.imprimir()

    val vecProducto = vec1 * vec2
    println("El producto componente a componente de los dos vectores")
    vecProducto.imprimir()

    val vecDivision = vec1 / vec2
    println("La división componente a componente de los dos vectores")
    vecDivision.imprimir()
}

```

Ejecución

```
11 6 7 4 10
9 10 7 7 7
Suma componente a componente de los dos vectores
20 16 14 11 17
La resta componente a componente de los dos vectores
2 -4 0 -3 3
El producto componente a componente de los dos vectores
99 60 49 28 70
La división componente a componente de los dos vectores
1 0 1 0 1
```

Ejercicio 163 - clase Vector con sobrecarga

```
// Clase Vector con sobrecarga del operador * para producto escalar con un
entero
class Vector {
    val arreglo = IntArray(5)

    fun cargar() {
        for (i in arreglo.indices)
            arreglo[i] = (Math.random() * 11 + 1).toInt()
    }

    fun imprimir() {
        for (elemento in arreglo)
            print("$elemento ")
        println()
    }

    operator fun times(valor: Int): Vector {
        val resultado = Vector()
        for (i in arreglo.indices)
            resultado.arreglo[i] = arreglo[i] * valor
        return resultado
    }
}

fun main(args: Array<String>) {
    val vec1 = Vector()
    vec1.cargar()
    vec1.imprimir()

    println("El producto de un vector con el número 10 es")
    val vecProductoEnt = vec1 * 10
    vecProductoEnt.imprimir()
}
```

Ejecución

```
4 8 4 1 2
El producto de un vector con el número 10 es
40 80 40 10 20
```

Ejercicio 164 - Arreglo de cinco enteros

```
class Vector {
    // Se declara un arreglo de 5 enteros (valores iniciales serán 0)
    val arreglo = IntArray(5)

    // Método para cargar el arreglo con números aleatorios entre 1 y 11
    fun cargar() {
        for(i in arreglo.indices)
            arreglo[i] = (Math.random() * 11 + 1).toInt()
    }

    // Método para imprimir los elementos del arreglo separados por espacio
    fun imprimir() {
        for(elemento in arreglo)
            print("$elemento ")
        println()
    }

    // Sobrecarga del operador ++ (incremento)
    operator fun inc(): Vector {
        // Se crea un nuevo objeto Vector
        var sumal = Vector()
        // Se copia cada valor del vector original sumándole 1
        for(i in arreglo.indices)
            sumal.arreglo[i] = arreglo[i] + 1
        return sumal // Se devuelve el nuevo vector incrementado
    }

    // Sobrecarga del operador -- (decremento)
    operator fun dec(): Vector {
        // Se crea un nuevo objeto Vector
        var restal = Vector()
        // Se copia cada valor del vector original restándole 1
        for(i in arreglo.indices)
            restal.arreglo[i] = arreglo[i] - 1
        return restal // Se devuelve el nuevo vector decrementado
    }
}
```

Ejercicio 165 - Comparación de objetos

```
// Comparación de objetos mediante el operador compareTo
class Persona(val nombre: String, val edad: Int) {
    fun imprimir() {
        println("Nombre: $nombre y tiene una edad de $edad")
    }
}
```

```

// Sobrecarga de compareTo para comparar edades
operator fun compareTo(per2: Persona): Int {
    return when {
        edad < per2.edad -> -1
        edad > per2.edad -> 1
        else -> 0
    }
}

fun main(parametro: Array<String>) {
    val personal = Persona("Juan", 30)
    personal.imprimir()
    val persona2 = Persona("Ana", 30)
    persona2.imprimir()

    println("Persona mayor")
    when {
        personal > persona2 -> personal.imprimir()
        personal < persona2 -> persona2.imprimir()
        else -> println("Tienen la misma edad")
    }
}

```

Ejecución

```

Nombre: Juan y tiene una edad de 30
Nombre: Ana y tiene una edad de 30
Persona mayor
Tienen la misma edad

```

Ejercicio 166 - class TaTeTi

```

// Clase para simular un tablero de Ta-Te-Ti utilizando operadores [] get y set
class TaTeTi {
    val tablero = IntArray(9)

    fun imprimir() {
        for (fila in 0..2) {
            for (columna in 0..2)
                print("${this[fila, columna]} ")
            println()
        }
        println()
    }

    // Asignar un valor a una posición del tablero
    operator fun set(fila: Int, columna: Int, valor: Int) {
        tablero[fila * 3 + columna] = valor
        imprimir()
    }
}

```

```

// Obtener el valor de una posición del tablero
operator fun get(fila: Int, columna: Int): Int {
    return tablero[fila * 3 + columna]
}

fun main(args: Array<String>) {
    val juego = TaTeTi()
    juego[0, 0] = 1
    juego[0, 2] = 2
    juego[2, 0] = 1
    juego[1, 2] = 2
    juego[1, 0] = 1
}

```

Ejecución

```

1 0 0
0 0 0
0 0 0

1 0 2
0 0 0
0 0 0

1 0 2
0 0 0
1 0 0

1 0 2
0 0 2
1 0 0

1 0 2
1 0 2
1 0 0

```

Ejercicio 167 - Clase Dados con invoke

```

// Clase Dados con operador invoke para acceder a valores como si fueran
función
class Dados {
    val arreglo = IntArray(10)

    fun tirar() {
        for (i in arreglo.indices)
            arreglo[i] = ((Math.random() * 6) + 1).toInt()
    }

    // Permite invocar al objeto como si fuera función: dados(0)
    operator fun invoke(nro: Int) = arreglo[nro]
}

fun main(args: Array<String>) {
    val dados = Dados()
    dados.tirar()
}

```

```

    println(dados(0))
    println(dados(1))
    for (i in 2..9)
        println(dados(i))
}

```

Ejecución

```

4
4
1
1
6
5
5
5
5
6

```

Ejercicio 168 - Clase vector con +=

```

// Clase Vector con sobrecarga del operador +=
class Vector {
    val arreglo = IntArray(5) { it }

    fun imprimir() {
        for (elemento in arreglo)
            print("$elemento ")
        println()
    }

    // Sobrecarga del operador += para sumar vectores componente a componente
    operator fun plusAssign(vec2: Vector) {
        for (i in arreglo.indices)
            arreglo[i] += vec2.arreglo[i]
    }
}

fun main(args: Array<String>) {
    val vec1 = Vector()
    vec1.imprimir()

    val vec2 = Vector()
    vec2.imprimir()

    vec1 += vec2
    vec1.imprimir()
}

```

Ejecución

```
0 1 2 3 4  
0 1 2 3 4  
0 2 4 6 8
```

Ejercicio 169 - Operador in

```
// Uso del operador in para verificar si un documento está en un curso  
data class Alumno(val documento: Int, val nombre: String)  
  
class Curso {  
    val alumnos = arrayOf(  
        Alumno(123456, "Marcos"),  
        Alumno(666666, "Ana"),  
        Alumno(777777, "Luis")  
    )  
  
    // Sobrecarga del operador 'in'  
    operator fun contains(documento: Int): Boolean {  
        return alumnos.any { documento == it.documento }  
    }  
}  
  
fun main(args: Array<String>) {  
    val curso1 = Curso()  
    if (123456 in curso1)  
        println("El alumno Marcos está inscripto en el curso")  
    else  
        println("El alumno Marcos no está inscripto en el curso")  
}
```

Ejecución

```
El alumno Marcos está inscripto en el curso
```

38. Funciones: número variable de parámetros

Ejercicio 170 - Función número variable

```
// Función que recibe un número variable de argumentos y los suma  
fun sumar(vararg numeros: Int): Int {  
    var suma = 0  
    for (elemento in numeros)  
        suma += elemento  
    return suma  
}
```

```

fun main(args: Array<String>) {
    val total = sumar(10, 20, 30, 40, 50)
    println(total)
}

```

Ejecución

150

Ejercicio 171 - enum y argumentos variables

```

// Uso de enum y argumentos variables en una función para sumar o promediar
enum class Operacion {
    SUMA,
    PROMEDIO
}

fun operar(tipoOperacion: Operacion, vararg arreglo: Int): Int {
    return when (tipoOperacion) {
        Operacion.SUMA -> arreglo.sum()
        Operacion.PROMEDIO -> arreglo.average().toInt()
    }
}

fun main(args: Array<String>) {
    val resultado1 = operar(Operacion.SUMA, 10, 20, 30)
    println("La suma es $resultado1")
    val resultado2 = operar(Operacion.PROMEDIO, 10, 20, 30)
    println("El promedio es $resultado2")
}

```

Ejecución

La suma es 60
El promedio es 20

Problema propuesto

Confeccionar una función que reciba una serie de edades y nos retorne la cantidad que son mayores o iguales a 18 (como mínimo se envía un entero a la función)

Ejercicio 172 - función cantidadMayores

```

// Función que recibe un número variable de argumentos enteros (`vararg`)
// llamados `edades`
// y retorna la cantidad de elementos mayores o iguales a 18.
fun cantidadMayores(vararg edades: Int) = edades.count { it >= 18 }
fun main(args: Array<String>) =
    println(cantidadMayores(3, 65, 32, 23, 2, 98, 23, 45, 15))

```

Ejecución

6

39. Colecciones - List y MutableList

Ejercicio 173 - Listas inmutables

```
// Uso de listas inmutables en Kotlin
fun main(args: Array<String>) {
    val lista1: List<String> = listOf(
        "lunes", "martes", "miércoles", "jueves",
        "viernes", "sábado", "domingo"
    )

    println("Imprimir la lista completa")
    println(lista1)

    println("Imprimir el primer elemento de la lista")
    println(lista1[0])
    println(lista1.first())

    println("Imprimir el último elemento de la lista")
    println(lista1.last())
    println(lista1[lista1.size - 1])

    println("Imprimir la cantidad de elementos de la lista")
    println(lista1.size)

    println("Recorrer la lista completa con un for")
    for (elemento in lista1)
        print("$elemento ")
    println()

    println("Imprimir el elemento y su posición")
    for (posicion in lista1.indices)
        print("[${posicion}] ${lista1[posicion]} ")
}
```

Ejecución

```
Imprimir la lista completa
[lunes, martes, miércoles, jueves, viernes, sábado, domingo]
Imprimir el primer elemento de la lista
lunes
lunes
Imprimir el último elemento de la lista
domingo
domingo
Imprimir la cantidad de elementos de la lista
7
Recorrer la lista completa con un for
lunes martes miércoles jueves viernes sábado domingo
Imprimir el elemento y su posición
[0]lunes [1]martes [2]miércoles [3]jueves [4]viernes [5]sábado [6]domingo
```

Ejercicio 174 - Lista entrada por teclado

```
// Crear una lista con entrada por teclado
fun cargar(): Int {
    print("Ingrese un entero: ")
    return readLine() !!.toInt()
}

fun main(args: Array<String>) {
    val lista1: List<Int> = List(5) { cargar() }
    println(lista1)
}
```

Ejecución

```
Ingrese un entero: 5
Ingrese un entero: 1
Ingrese un entero: 3
Ingrese un entero: 9
Ingrese un entero: 6
[5, 1, 3, 9, 6]
```

Ejercicio 175 - Operaciones con una lista inmutable

```
// Operaciones con una lista mutable de edades
fun main(args: Array<String>) {
    val edades: MutableList<Int> = mutableListOf(23, 67, 12, 35, 12)
    println("Lista de edades")
    println(edades)

    edades[0] = 60
    println("Lista completa después de modificar la primera edad")
    println(edades)

    println("Primera edad almacenada en la lista")
```

```

println(edades[0])

println("Promedio de edades")
println(edades.average())

println("Agregamos una edad al final de la lista:")
edades.add(50)
println(edades)

println("Agregamos una edad al principio de la lista:")
edades.add(0, 17)
println(edades)

println("Eliminamos la primer edad de la lista:")
edades.removeAt(0)
println(edades)

print("Cantidad de personas mayores de edad:")
val cant = edades.count { it >= 18 }
println(cant)

edades.removeAll { it == 12 }
println("Lista después de borrar las edades con valor 12")
println(edades)

edades.clear()
println("Lista luego de borrar todos los elementos")
println(edades)

if (edades.isEmpty())
    println("No hay edades almacenadas en la lista")
}

```

Ejecución

```

Lista de edades
[23, 67, 12, 35, 12]
Lista completa después de modificar la primer edad
[60, 67, 12, 35, 12]
Primera edad almacenada en la lista
60
Promedio de edades
37.2
Agregamos una edad al final de la lista:
[60, 67, 12, 35, 12, 50]
Agregamos una edad al principio de la lista:
[17, 60, 67, 12, 35, 12, 50]
Eliminamos la primer edad de la lista:
[60, 67, 12, 35, 12, 50]
Cantidad de personas mayores de edad:4
Lista después de borrar las edades con valor 12
[60, 67, 35, 50]
Lista luego de borrar todos los elementos
[]
No hay edades almacenadas en la lista

```

Ejercicio 176 - Lista de enteros aleatorios

```
// Lista de enteros aleatorios y operaciones de filtrado
fun main(args: Array<String>) {
    val lista: MutableList<Int> = MutableList(20) { ((Math.random() * 6) +
1).toInt() }
    println("Lista completa")
    println(lista)

    val cant = lista.count { it == 1 }
    println("Cantidad de elementos con el valor 1: $cant")

    lista.removeAll { it == 6 }
    println("Lista luego de borrar los elementos con el valor 6")
    println(lista)
}
```

Ejecución

```
Lista completa
[6, 1, 1, 3, 2, 1, 3, 5, 4, 2, 5, 5, 6, 4, 3, 5, 2, 3, 6, 3]
Cantidad de elementos con el valor 1: 3
Lista luego de borrar los elementos con el valor 6
[1, 1, 3, 2, 1, 3, 5, 4, 2, 5, 5, 4, 3, 5, 2, 3, 3]
```

Ejercicio 177 - Lista de objetos persona y conteo

```
// Lista de objetos Persona y conteo de mayores de edad
class Persona(var nombre: String, var edad: Int) {
    fun imprimir() {
        println("Nombre: $nombre y tiene una edad de $edad")
    }

    fun esMayorEdad() {
        if (edad >= 18)
            println("Es mayor de edad $nombre")
        else
            println("No es mayor de edad $nombre")
    }
}

fun main(args: Array<String>) {
    val personas: MutableList<Persona> = mutableListOf(
        Persona("Juan", 22),
        Persona("Ana", 19),
        Persona("Marcos", 12)
    )

    println("Listado de todas las personas")
    personas.forEach { it.imprimir() }

    val cant = personas.count { it.edad >= 18 }
    println("La cantidad de personas mayores de edad es $cant")
}
```

Ejecución

```
Listado de todas las personas
Nombre: Juan y tiene una edad de 22
Nombre: Ana y tiene una edad de 19
Nombre: Marcos y tiene una edad de 12
La cantidad de personas mayores de edad es 2
```

Problemas propuestos

Crear una lista inmutable de 100 elementos enteros con valores aleatorios comprendidos entre 0 y 20.

contar cuantos hay comprendidos entre 1 y 4, 5 y 8 y cuantos entre 10 y 13.

Imprimir si el valor 20 está presente en la lista.

Confeccionar una clase que represente un Empleado. Definir como propiedades su nombre y su sueldo.

Definir una lista mutable con tres empleados.

Imprimir los datos de los empleados.

Calcular cuanto gasta la empresa en sueldos.

Agregar un nuevo empleado a la lista y calcular nuevamente el gasto en sueldos.

Cargar por teclado y almacenar en una lista inmutable las alturas de 5 personas (valores Float)

Obtener el promedio de las mismas. Contar cuántas personas son más altas que el promedio y cuántas más bajas.

Ejercicio 178 - Lista de cien elementos aleatorios

```
fun main(args: Array<String>) {
    // Crea una lista de 100 elementos, cada uno con un número aleatorio entre
    0 y 20 (inclusive)
    val lista1 = List(100, { ((Math.random() * 21)).toInt() })

    // Imprime la lista completa
    println(lista1)

    // Variables para contar cuántos elementos están en ciertos rangos
    var cant1 = 0 // Para valores entre 1 y 4
    var cant2 = 0 // Para valores entre 5 y 8
    var cant3 = 0 // Para valores entre 10 y 13

    // Recorre cada elemento de la lista y aumenta los contadores según el
    valor
    lista1.forEach {
        when (it) {
            in 1..4 -> cant1++
            in 5..8 -> cant2++
            in 10..13 -> cant3++
        }
    }
}
```

```

}

// Imprime los resultados de los conteos
println("Cantidad de valores comprendidos entre 1..4: $cant1")
println("Cantidad de valores comprendidos entre 5..8: $cant2")

// ⚠ ERROR: aquí debería imprimirse `cant3`, no `cant1`
    println("Cantidad de valores comprendidos entre 10..13: $cant1") // ←
estos están mal

// CORREGIDO:
// println("Cantidad de valores comprendidos entre 10..13: $cant3")

// Verifica si en la lista hay algún 20
if (list1.contains(20))
    println("La lista contiene el 20")
else
    println("La lista no contiene el 20")
}

```

Ejecución

```

[16, 2, 3, 14, 6, 19, 6, 6, 13, 11, 18, 1, 15, 14, 16, 7, 2, 5, 10, 4, 13, 5, 10, 8, 5, 1, 11, 14, 6, 2, 14, 9, 13, 18,
Cantidad de valores comprendidos entre 1..4: 18
Cantidad de valores comprendidos entre 5..8: 26
Cantidad de valores comprendidos entre 10..13: 18
La lista contiene el 20

7, 12, 9, 13, 5, 20, 4, 18, 4, 16, 1, 8, 20, 11, 20, 16, 18, 18, 16, 13, 15, 4, 20, 0, 19, 2, 8, 5, 7, 13, 7, 20, 18, 13, 4,
13, 10, 5, 5, 0, 7, 7, 20, 17, 20, 9, 5, 2, 9, 5, 9, 18, 8, 3, 14, 17, 14, 12, 3, 5, 18, 7, 3]

```

Ejercicio 179 - clase Empleado nombre y sueldo

```

// Definimos la clase Empleado con dos propiedades: nombre (String) y sueldo
(Double)
class Empleado(var nombre: String, var sueldo: Double) {

    // Función que imprime la información del empleado
    fun imprimir() {
        println("$nombre tiene un sueldo de $sueldo")
    }
}

// Función que recibe una lista mutable de empleados y calcula la suma total
de sus sueldos
fun calcularGastos(empleados: MutableList<Empleado>) {
    // sumByDouble suma los sueldos de todos los empleados en la lista
}

```

```

    val suma = empleados.sumByDouble { it.sueldo }
    println("Total de gastos de la empresa: $suma")
}

fun main(args: Array<String>) {
    // Creamos una lista mutable de empleados, con tres empleados iniciales
    val empleados: MutableList<Empleado> = mutableListOf(
        Empleado("Juan", 2000.0),
        Empleado("Ana", 3500.0),
        Empleado("Carlos", 1500.0)
    )

    // Imprimimos los datos de cada empleado
    empleados.forEach { it.imprimir() }

    // Calculamos e imprimimos el gasto total en sueldos
    calcularGastos(empleados)

    // Agregamos un nuevo empleado a la lista
    empleados.add(Empleado("Marcos", 3000.0))

    // Indicamos que se recalcularán los gastos con el nuevo empleado
    println("Gastos luego de ingresar un nuevo empleado que cobra 3000")

    // Volvemos a calcular los gastos con el nuevo empleado incluido
    calcularGastos(empleados)
}

```

Ejecución

```

Juan tiene un sueldo de 2000.0
Ana tiene un sueldo de 3500.0
Carlos tiene un sueldo de 1500.0
Total de gastos de la empresa: 7000.0
Gastos luego de ingresar un nuevo empleado que cobra 3000
Total de gastos de la empresa: 10000.0

```

Ejercicio 180 - Función cargar

```

// Función para pedir al usuario que ingrese una altura (valor tipo Float)
fun cargar(): Float {
    print("Ingrese la altura de la persona (Ej. 1.92): ")
    val valor = readln().toFloat() // Lee la entrada del usuario y la
    convierte a Float
    return valor // Retorna la altura ingresada
}

fun main(args: Array<String>) {
    // Creamos una lista de 5 alturas, llamando a la función cargar() 5 veces
    val lista1: List<Float> = List(5) { cargar() }

    // Calculamos la altura promedio con la función average()
    val promedio = lista1.average()
    println("La altura promedio de las personas es $promedio")
}

```

```

// Contamos cuántas alturas son mayores al promedio
val altos = lista1.count { it > promedio }

// Contamos cuántas alturas son menores al promedio
val bajos = lista1.count { it < promedio }

// Mostramos la cantidad de personas más altas y más bajas que el promedio
println("La cantidad de personas más altas al promedio es: $altos")
println("La cantidad de personas más bajas al promedio es: $bajos")
}

```

Ejecución

```

Ingrese la altura de la persona (Ej. 1.92): 1.90
Ingrese la altura de la persona (Ej. 1.92): 1.81
Ingrese la altura de la persona (Ej. 1.92): 1.78
Ingrese la altura de la persona (Ej. 1.92): 1.60
Ingrese la altura de la persona (Ej. 1.92): 1.58
La altura promedio de las personas es 1.733999991416931
La cantidad de personas más altas al promedio es: 3
La cantidad de personas más bajas al promedio es: 2

```

40. Colecciones - Map y MutableMap

Ejercicio 181 - Map en kotlin

```

// Uso de Map en Kotlin con datos de países y habitantes
fun main(args: Array<String>) {
    val paises: Map<String, Int> = mapOf(
        "argentina" to 40000000,
        "españa" to 46000000,
        "uruguay" to 3400000
    )

    println("Listado completo del Map")
    println(paises)

    for ((clave, valor) in paises)
        println("Para la clave $clave tenemos almacenado $valor")

    println("La cantidad de elementos del mapa es ${paises.size}")

    val cantHabitantes1: Int? = paises["argentina"]
    if (cantHabitantes1 != null)
        println("La cantidad de habitantes de argentina es ${cantHabitantes1}")

    val cantHabitantes2: Int? = paises["brasil"]
    if (cantHabitantes2 != null)
        println("La cantidad de habitantes de brasil es ${cantHabitantes2}")
    else

```

```

    println("brasil no se encuentra cargado en el Map")

    var suma = 0
    paises.forEach { suma += it.value }
    println("Cantidad total de habitantes de todos los países es $suma")
}

```

Ejecución

```

Listado completo del Map
{argentina=40000000, españa=46000000, uruguay=3400000}
Para la clave argentina tenemos almacenado 40000000
Para la clave españa tenemos almacenado 46000000
Para la clave uruguay tenemos almacenado 3400000
La cantidad de elementos del mapa es 3
La cantidad de habitantes de argentina es 40000000
brasil no se encuentra cargado en el Map
Cantidad total de habitantes de todos los países es 89400000

```

Ejercicio 182 - Funciones para trabajar con Map

```

// Funciones para trabajar con Map de productos y sus precios
fun imprimir(productos: Map<String, Float>) {
    for ((clave, valor) in productos)
        println("$clave tiene un precio $valor")
    println()
}

fun cantidadPrecioMayor20(productos: Map<String, Float>) {
    val cant = productos.count { it.value > 20 }
    println("Cantidad de productos con un precio superior a 20: $cant")
}

fun main(args: Array<String>) {
    val productos: Map<String, Float> = mapOf(
        "papas" to 12.5f,
        "manzanas" to 26f,
        "peras" to 31f,
        "mandarinas" to 15f,
        "pomelos" to 28f
    )
    imprimir(productos)
    cantidadPrecioMayor20(productos)
}

```

Ejecución

```
papas tiene un precio 12.5
manzanas tiene un precio 26.0
peras tiene un precio 31.0
mandarinas tiene un precio 15.0
pomelos tiene un precio 28.0

Cantidad de productos con un precio superior a 20: 3
```

Ejercicio 183 - Diccionario castellano-inglés

```
// Diccionario castellano-inglés usando MutableMap
fun cargar(diccionario: MutableMap<String, String>) {
    do {
        print("Ingrese palabra en castellano: ")
        val palCastellano = readLine()!!
        print("Ingrese palabra en inglés: ")
        val palIngles = readLine()!!
        diccionario[palIngles] = palCastellano

        print("¿Continúa cargando otra palabra en el diccionario? (si/no): ")
        val fin = readLine()!!
    } while (fin == "si")
}

fun listado(diccionario: MutableMap<String, String>) {
    println("Diccionario completo:")
    for ((ingles, castellano) in diccionario)
        println("$ingles: $castellano")
    println()
}

fun consultaIngles(diccionario: MutableMap<String, String>) {
    print("Ingrese una palabra en inglés para verificar su traducción: ")
    val ingles = readLine()
    if (ingles in diccionario)
        println("La traducción en castellano es ${diccionario[ingles]}")
    else
        println("No existe esa palabra en el diccionario")
}

fun main(args: Array<String>) {
    val diccionario: MutableMap<String, String> = mutableMapOf()
    cargar(diccionario)
    listado(diccionario)
    consultaIngles(diccionario)
}
```

Ejecución

```
Ingrese palabra en castellano: Lobo
Ingrese palabra en inglés: Wolf
¿Continúa cargando otra palabra en el diccionario? (si/no): no
Diccionario completo:
Wolf: Lobo

Ingrese una palabra en inglés para verificar su traducción: Wolf
La traducción en castellano es Lobo
```

Ejercicio 184 - Mapa de productos

```
// Mapa de productos usando data class para nombre, precio y cantidad
data class Producto(val nombre: String, val precio: Float, val cantidad: Int)

fun cargar(productos: MutableMap<Int, Producto>) {
    productos[1] = Producto("Papas", 13.15f, 200)
    productos[15] = Producto("Manzanas", 20f, 0)
    productos[20] = Producto("Peras", 3.50f, 0)
}

fun listadoCompleto(productos: MutableMap<Int, Producto>) {
    println("Listado completo de productos")
    for ((codigo, producto) in productos)
        println("Código: $codigo Descripción: ${producto.nombre} Precio: ${producto.precio} Stock Actual: ${producto.cantidad}")
    println()
}

fun consultaProducto(productos: MutableMap<Int, Producto>) {
    print("Ingrese el código de un producto: ")
    val codigo = readLine()!!.toInt()
    if (codigo in productos) {
        val producto = productos[codigo]
        println("Nombre: ${producto?.nombre} Precio: ${producto?.precio} Stock: ${producto?.cantidad}")
    } else {
        println("No existe un producto con dicho código")
    }
}

fun sinStock(productos: MutableMap<Int, Producto>) {
    val cant = productos.count { it.value.cantidad == 0 }
    println("Cantidad de artículos sin stock: $cant")
}

fun main(args: Array<String>) {
    val productos: MutableMap<Int, Producto> = mutableMapOf()
    cargar(productos)
    listadoCompleto(productos)
    consultaProducto(productos)
    sinStock(productos)
}
```

```
}
```

Ejecución

```
Listado completo de productos
Código: 1 Descripción: Papas Precio: 13.15 Stock Actual: 200
Código: 15 Descripción: Manzanas Precio: 20.0 Stock Actual: 0
Código: 20 Descripción: Peras Precio: 3.5 Stock Actual: 0

Ingrese el código de un producto: 15
Nombre: Manzanas Precio: 20.0 Stock: 0
Cantidad de artículos sin stock: 2
```

Ejercicio 185 - Diccionario de alumnos

```
// Diccionario de alumnos con materias y notas usando Map y List
data class Materia(val nombre: String, val nota: Int)

fun cargar(alumnos: MutableMap<Int, MutableList<Materia>>) {
    print("¿Cuántos alumnos cargará?: ")
    val cant = readLine()!!.toInt()

    for (i in 1..cant) {
        print("Ingrese DNI: ")
        val dni = readLine()!!.toInt()
        val listaMaterias = mutableListOf<Materia>()

        do {
            print("Ingrese materia del alumno: ")
            val nombre = readLine()!!
            print("Ingrese nota: ")
            val nota = readLine()!!.toInt()
            listaMaterias.add(Materia(nombre, nota))
            print("¿Ingresar otra nota? (si/no): ")
            val opcion = readLine()!!
        } while (opcion == "si")

        alumnos[dni] = listaMaterias
    }
}

fun imprimir(alumnos: MutableMap<Int, MutableList<Materia>>) {
    for ((dni, listaMaterias) in alumnos) {
        println("DNI del alumno: $dni")
        for (materia in listaMaterias) {
            println("Materia: ${materia.nombre}")
            println("Nota: ${materia.nota}")
        }
        println()
    }
}

fun consultaPorDni(alumnos: MutableMap<Int, MutableList<Materia>>) {
```

```

print("Ingrese el DNI del alumno a consultar: ")
val dni = readLine()!!.toInt()

if (dni in alumnos) {
    println("Cursa las siguientes materias:")
    val lista = alumnos[dni]
    if (lista != null) {
        for ((nombre, nota) in lista) {
            println("Nombre de materia: $nombre")
            println("Nota: $nota")
        }
    }
}
}

fun main(args: Array<String>) {
    val alumnos: MutableMap<Int, MutableList<Materia>> = mutableMapOf()
    cargar(alumnos)
    imprimir(alumnos)
    consultaPorDni(alumnos)
}

```

Ejecución

```

¿Cuántos alumnos cargará?: 2
Ingrese DNI: 9473727
Ingrese materia del alumno: Inglés
Ingrese nota: 90
¿Ingresar otra nota? (si/no): si
Ingrese materia del alumno: Programación
Ingrese nota: 84
¿Ingresar otra nota? (si/no): no
Ingrese DNI: 74363643
Ingrese materia del alumno: Big data
Ingrese nota: 70
¿Ingresar otra nota? (si/no): no
DNI del alumno: 9473727
Materia: Inglés
Nota: 90
Materia: Programación
Nota: 84

DNI del alumno: 74363643
Materia: Big data
Nota: 70

Ingrese el DNI del alumno a consultar: 9473727
Cursa las siguientes materias:
Nombre de materia: Inglés
Nota: 90
Nombre de materia: Programación
Nota: 84

```

Problema propuesto

Confeccionar una agenda. Utilizar un MutableMap cuya clave sea de la clase Fecha:

```
data class Fecha(val dia: Int, val mes: Int, val año: Int)
```

Como valor en el mapa almacenar un String.

Implementar las siguientes funciones:

- 1) Carga de datos en la agenda.
- 2) Listado completo de la agenda.
- 3) Consulta de una fecha.

Ejercicio 186 - Clase de datos para fecha

```
// Definición de una clase de datos para representar una fecha
data class Fecha(val dia: Int, val mes: Int, val año: Int)

// Función que permite ingresar varias fechas con actividades
fun cargar(agenda: MutableMap<Fecha, String>) {
    do {
        println("Ingrese fecha")
        print("Ingrese el día:")
        val dia = readln().toInt()
        print("Ingrese el mes:")
        val mes = readln().toInt()
        print("Ingrese el año:")
        val año = readln().toInt()

        print("Ingrese todas las actividades para ese día:")
        val actividades = readln()

        // Se guarda en el mapa la actividad asociada a la fecha
        agenda[Fecha(dia, mes, año)] = actividades

        // Pregunta si desea ingresar otra fecha
        print("Ingrese otra fecha (si/no):")
        val opcion = readln()
    } while (opcion.lowercase() == "si")
}

// Función para mostrar toda la agenda
fun imprimir(agenda: MutableMap<Fecha, String>) {
    for ((fecha, actividad) in agenda) {
        println("Fecha ${fecha.dia}/${fecha.mes}/${fecha.año}")
        println("Actividades: $actividad")
        println()
    }
}

// Función que permite consultar actividades por una fecha específica
fun consultaFecha(agenda: MutableMap<Fecha, String>) {
    println("Ingrese una fecha a consultar")
    print("Ingrese el día:")
    val dia = readln().toInt()
    print("Ingrese el mes:")
    val mes = readln().toInt()
    print("Ingrese el año:")
```

```

val año = readln().toInt()

val fecha = Fecha(dia, mes, año)

// Consulta si la fecha existe en la agenda
if (fecha in agenda)
    println("Actividades: ${agenda[fecha]}")
else
    println("No existen actividades registradas para dicha fecha")
}

// Función principal
fun main(args: Array<String>) {
    val agenda: MutableMap<Fecha, String> = mutableMapOf()

    cargar(agenda)          // Cargar datos
    imprimir(agenda)        // Mostrar todos los datos
    consultaFecha(agenda)   // Consultar una fecha puntual
}

```

Ejecución

```

Ingrese fecha
Ingrese el día:15
Ingrese el mes:08
Ingrese el año:2024
Ingrese todas las actividades para ese dia:Correr, comer, estudiar, bañarme, dormir
Ingrese otra fecha (si/no):np
Fecha 15/8/2024
Actividades: Correr, comer, estudiar, bañarme, dormir

Ingrese una fecha a consultar
Ingrese el dia:15
Ingrese el mes:08
Ingrese el año:2024
Actividades: Correr, comer, estudiar, bañarme, dormir

```

41. Colecciones - Set y MutableSet

Ejercicio 187 - Operaciones con conjuntos mutables

```

// Operaciones con conjuntos mutables (MutableSet)
fun main(args: Array<String>) {
    val conjunto1: MutableSet<Int> = mutableSetOf(2, 7, 20, 150, 3)

    println("Listado completo del conjunto")
    for (elemento in conjunto1)
        print("$elemento ")
    println()

    println("Cantidad de elementos del conjunto: ${conjunto1.size}")

    conjunto1.add(500)
    println("Después de agregar el 500:")
    println(conjunto1)
}

```

```

conjunto1.add(500)
println("Después de intentar agregar el 500 nuevamente (no se repite):")
println(conjunto1)

if (500 in conjunto1)
    println("El 500 está almacenado en el conjunto")
else
    println("El 500 no está almacenado en el conjunto")

println("Eliminamos el elemento 500 del conjunto")
conjunto1.remove(500)

if (500 in conjunto1)
    println("El 500 está almacenado en el conjunto")
else
    println("El 500 no está almacenado en el conjunto")

val cant = conjunto1.count { it >= 10 }
    println("Cantidad de elementos con valores superiores o igual a 10:
$cant")
}

```

Ejecución

```

Listado completo del conjunto
2 7 20 150 3
Cantidad de elementos del conjunto: 5
Después de agregar el 500:
[2, 7, 20, 150, 3, 500]
Después de intentar agregar el 500 nuevamente (no se repite):
[2, 7, 20, 150, 3, 500]
El 500 está almacenado en el conjunto
Eliminamos el elemento 500 del conjunto
El 500 no está almacenado en el conjunto
Cantidad de elementos con valores superiores o igual a 10: 2

```

Ejercicio 188 - Verificación de fechas

```

// Verificación de fechas usando Set de objetos data class
data class Fecha(val dia: Int, val mes: Int, val año: Int)

fun main(args: Array<String>) {
    val feriados: Set<Fecha> = setOf(
        Fecha(1, 1, 2017),
        Fecha(25, 12, 2017)
    )

    println("Ingrese una fecha")
    print("Día: ")
    val dia = readLine()!!.toInt()

```

```

print("Mes: ")
val mes = readLine()!!.toInt()
print("Año: ")
val año = readLine()!!.toInt()

if (Fecha(dia, mes, año) in feriados)
    println("La fecha ingresada es feriado")
else
    println("La fecha ingresada no es feriado")
}

```

Ejecución

```

Ingrese una fecha
Día: 08
Mes: 02
Año: 2002
La fecha ingresada no es feriado

```

Problema propuesto

Definir un MutableSet y almacenar 6 valores aleatorios comprendidos entre 1 y 50. El objeto de tipo MutableSet representa un cartón de lotería. Comenzar a generar valores aleatorios (comprendidos entre 1 y 5) todos distintos y detenerse cuando salgan todos los que contiene el cartón de lotería. Mostrar cuantos números tuvieron que sortearse hasta que se completó el cartón.

Ejercicio 189 - MutableSet almacena seis valores

```

// Esta función genera un cartón de lotería con 6 números aleatorios únicos
entre 1 y 50
fun generarCarton(): Set<Int> {
    return (1..50).shuffled().take(6).toSet()
    // (1..50) → genera una secuencia del 1 al 50
    // shuffled() → desordena aleatoriamente los elementos
    // take(6) → toma los primeros 6 elementos
    // toSet() → convierte la lista en un conjunto (sin duplicados)
}

// Esta función genera una lista con los números del bolillero del 1 al 50 en
orden aleatorio
fun generarBolillero(): List<Int> {
    return (1..50).shuffled()
    // Aquí se genera una lista con los números del 1 al 50 desordenados
}

// Esta función verifica cuántas bolillas se deben sacar hasta que el cartón
acierte las 6
fun verificarTriunfo(carton: Set<Int>, bolillero: List<Int>) {
    var aciertos = 0           // Contador de aciertos (coincidencias con el
    cartón)
    var cantBolillas = 0       // Cantidad de bolillas sacadas
}

```

```

// Se recorre cada bolilla extraída
for (bolilla in bolillero) {
    cantBolillas++
    if (bolilla in carton) { // Si el número de la bolilla está en el
cartón
        aciertos++
        if (aciertos == 6) break // Si ya acertó las 6, se detiene el
ciclo
    }
}

// Se muestra cuántas bolillas se necesitaron para ganar
println("Se sacaron $cantBolillas bolillas hasta que el cartón ganó.")
}

fun main() {
    // Generamos el cartón de 6 números
    val carton = generarCarton()
    println("Cartón de lotería generado:")
    println(carton)

    // Generamos el bolillero (lista de 50 números desordenados)
    val bolillero = generarBolillero()
    println("Números del bolillero:")
    println(bolillero)

    // Verificamos cuántas bolillas se sacaron hasta ganar
    verificarTriunfo(carton, bolillero)
}

```

Ejecución

```

Cartón de lotería generado:
[23, 29, 19, 36, 33, 34]
Números del bolillero:
[19, 3, 7, 23, 33, 34, 48, 42, 27, 16, 30, 25, 45, 17, 39, 20, 15, 10, 35, 38, 40, 37, 14, 31, 50, 2, 4, 5, 18,
Se sacaron 50 bolillas hasta que el cartón ganó.

```

```
26, 46, 49, 8, 13, 11, 9, 47, 28, 12, 6, 43, 41, 22, 36, 32, 24, 1, 44, 21, 29]
```

42. Package e Import

Ejercicio 190 - Archivo con funciones

```
// Archivo con funciones y uso en el mismo archivo (sin paquete separado)
```

```

fun sumar(valor1: Int, valor2: Int) = valor1 + valor2
fun restar(valor1: Int, valor2: Int) = valor1 - valor2

fun main(args: Array<String>) {
    val su = sumar(5, 7)
    println("La suma es $su")

    val re = restar(10, 3)
    println("La resta es $re")
}

```

Ejecución

```

La suma es 12
La resta es 7

```

Problema propuesto

Crear un paquete llamado entradateclado. Dentro del paquete crear un archivo llamado entrada.kt y definir las siguientes funciones:

```

package entradateclado

fun retornarInt(mensaje: String): Int {
    print(mensaje)
    return readln().toInt()
}

fun retornarDouble(mensaje: String): Double {
    print(mensaje)
    return readln().toDouble()
}

fun retornarFloat(mensaje: String): Float {
    print(mensaje)
    return readln().toFloat()
}

```

En el programa principal (Principal.tk) importar el paquete entradateclado y llamar a varias de sus funciones.

Ejercicio 191 - Cartón de lotería

```

// Esta función genera un cartón de lotería con 6 números aleatorios y únicos
// del 1 al 50
fun generarCarton(): Set<Int> {
    // Se crea una lista del 1 al 50, se mezcla aleatoriamente y se toman los
    // primeros 6 números
    return (1..50).shuffled().take(6).toSet()
}

```

```

// Esta función genera el bolillero (los 50 números del sorteo) en orden
aleatorio
fun generarBolillero(): List<Int> {
    // Se genera una lista del 1 al 50 y se mezcla aleatoriamente (como
extraer bolillas una a una)
    return (1..50).shuffled()
}

// Esta función simula el proceso de sacar bolillas del bolillero hasta que
se acierten los 6 números del cartón
fun verificarTriunfo(carton: Set<Int>, bolillero: List<Int>) {
    var aciertos = 0           // Contador de aciertos del cartón
    var cantBolillas = 0       // Contador de cuántas bolillas se han sacado

    // Recorremos el bolillero (los números extraídos en orden aleatorio)
    for (bolilla in bolillero) {
        cantBolillas++          // Cada número contado como una bolilla
extraída

        // Si la bolilla está en el cartón, se considera un acierto
        if (bolilla in carton) {
            aciertos++
            // Si se logran 6 aciertos, se detiene el ciclo
            if (aciertos == 6) break
        }
    }

    // Mostramos cuántas bolillas se necesitaron para ganar
    println("Se sacaron $cantBolillas bolillas hasta que el cartón ganó.")
}

fun main() {
    // Se genera el cartón de lotería y se imprime
    val carton = generarCarton()
    println("Cartón de lotería generado:")
    println(carton)

    // Se genera el bolillero mezclado y se imprime
    val bolillero = generarBolillero()
    println("Números del bolillero:")
    println(bolillero)

    // Se simula el juego y se verifica en cuántos intentos se gana
    verificarTriunfo(carton, bolillero)
}

```

Ejecución

```
Cartón de lotería generado:  
[12, 6, 33, 37, 7, 47]  
Números del bolillero:  
[41, 4, 31, 44, 34, 32, 47, 2, 18, 27, 1, 43, 40, 21, 46, 50, 16, 8, 7, 3, 12, 48, 38, 24, 5, 6, 20, 39, 45,  
Se sacaron 48 bolillas hasta que el cartón ganó.
```

```
49, 36, 23, 9, 14, 13, 15, 17, 37, 19, 25, 42, 26, 28, 30, 11, 10, 35, 33, 29, 22]
```

43. Corrutinas

Ejercicio 192 - Corrutinas y retardos

```
// Importamos las librerías necesarias para trabajar con corrutinas y  
retardos  
import kotlinx.coroutines.GlobalScope  
import kotlinx.coroutines.delay  
import kotlinx.coroutines.launch  
  
fun main(args: Array<String>) {  
    // Iniciamos una corriputina en el ámbito global (GlobalScope)  
    GlobalScope.launch {  
        // Imprime los números del 1 al 10, con una pausa de 1 segundo entre  
        // cada número  
        for (x in 1..10) {  
            print("$x ")          // Imprime el número actual  
            delay(1000)           // Suspende la corriputina durante 1000  
            milisegundos (1 segundo)  
        }  
    }  
  
    // Iniciamos otra corriputina simultáneamente  
    GlobalScope.launch {  
        // Imprime los números del 11 al 20, también con 1 segundo de pausa  
        // entre cada uno  
        for (x in 11..20) {  
            print("$x ")          // Imprime el número actual  
            delay(1000)           // Suspende esta corriputina por 1 segundo  
        }  
    }  
  
    // Esta línea mantiene el programa en ejecución hasta que el usuario  
    presione Enter  
    // Si no estuviera, el programa terminaría antes de que las corrutinas  
    completaran su ejecución
```

```
    readLine()
}
```

Problema propuesto

Confeccionar una aplicación que en su hilo principal se genere un valor aleatorio entre 1 y 100:

```
fun main(args: Array<String>) {
    val adivina = Random.nextInt(1, 100)
```

Luego crear una corutina donde la misma debe adivinar el número aleatorio generado en el hilo principal. Se debe generar un numero aleatorio, la primera vez entre 1 y 100, luego verificar si el número aleatorio a adivinar del hilo principal es mayor o menor. Si es igual mostrar que ganó y si no detenerse por 500 milisegundos y proceder a repetir la generación de otro número aleatorio pero ahora acotado a la respuesta de si era mayor o menor.

Repetir el proceso hasta que gane.

Ejercicio 193 - Corrutinas y retardos 2

```
// Importamos las librerías necesarias para trabajar con corrutinas y
retardos
import kotlinx.coroutines.GlobalScope
import kotlinx.coroutines.delay
import kotlinx.coroutines.launch

fun main(args: Array<String>) {
    // Iniciamos una corutina en el ámbito global (GlobalScope)
    GlobalScope.launch {
        // Imprime los números del 1 al 10, con una pausa de 1 segundo entre
        // cada número
        for (x in 1..10) {
            print("$x ")          // Imprime el número actual
            delay(1000)           // Suspende la corutina durante 1000
            // milisegundos (1 segundo)
        }
    }

    // Iniciamos otra corutina simultáneamente
    GlobalScope.launch {
        // Imprime los números del 11 al 20, también con 1 segundo de pausa
        // entre cada uno
        for (x in 11..20) {
            print("$x ")          // Imprime el número actual
            delay(1000)           // Suspende esta corutina por 1 segundo
        }
    }
}
```

```
// Esta línea mantiene el programa en ejecución hasta que el usuario
presione Enter
// Si no estuviera, el programa terminaría antes de que las corrutinas
completén su ejecución
    readLine()
}
```

44. Bibliografía

<https://www.tutorialesprogramacionya.com/kotlinya/index.php?inicio=0>