# BUCLES

1. Crea un menú que imprimeixi per consola les opcions:

MENU: 1) seguir al programa // 2) seguir al programa // 3) sortir

S’ha de mostrar el menú de nou mentre l’usuari premi 1 o 2, si selecciona 3, el programa ha d’acabar amb un missatge de “Adeu!!”

1. Imprimeix nombres entre 1 i 15 de la següent manera (els nombres han d’estar en un array):
   1. De 1 a 15 🡪 (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15)
   2. De 15 a 1 🡪 (15,14,13,12,11,10,9,8,7,6,5,4,3,2,1)
   3. De 1 a 15 només els parells 🡪 (2,4,6,8,10,12,14)
   4. De 1 a 15 de 5 en 5 🡪 (1, 5, 10, 15)

# FUNCIONS

1. Programa amb una funció que li passem un nombre com a paràmetre i imprimeix a consola (ex num 5):
   * + Iniciant el compte enrere
     + 5
     + 4
     + 3
     + 2
     + 1
     + 0
     + Compte enrere acabat
2. Fes un programa amb 4 funcions que han de fer el següent (el programa ha de cridar a les 4 funcions):
   1. Funció sense paràmetres que imprimeixi per consola “soc una funció”
   2. Funció que rebi un array de 5 nombres com a paràmetre, **retorni** l’índex del nombre més gran i imprimeixi nombre en si. El programa principal també haurà d’imprimir el nombre del array mitjançant el return de la funció.
   3. Funció que rebi un array de 5 noms com a paràmetre i **retoni** l’índex del nom més llarg (el programa “principal” haurà d’imprimir per consola el nom)
   4. Funció que rebi dos nombres i retorni un array amb tots els nombres compresos entre ells. El programa principal ha de mostrar aquest array per consola.

# POO

**ELECTRODOMESTICS**

Crearemos una clase llamada **Electrodomestico** con las siguientes características:

* Sus atributos son precio base, color, consumo energético (letras entre A y F). También precio Final.
* Los colores disponibles son blanco, negro, rojo, azul y gris. No importa si el nombre está en mayúsculas o en minúsculas.
* Los métodos que implementara serán:
  + Getters / Setters necesarios
  + **comprobarConsumoEnergetico(letra)**: comprueba que la letra es correcta, sino es correcta setea el consumo energético a F.
  + **comprobarColor(String color)**: comprueba que el color es correcto, sino lo es usa el color por defecto (blanco).
  + **precioFinal()**: según el consumo energético aumenta el precio final. Esta es la lista de precios (la cual debe usarse mediante un objeto tipo **Map()**):

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Map>

|  |  |
| --- | --- |
| **Letra** | **Precio** |
| A | 100 € |
| B | 80 € |
| C | 60 € |
| D | 50 € |
| E | 30 € |
| F | 10 € |
|  |  |

Crearemos una clase llamada **Lavadora** con las siguientes características:

* Su atributo propio es **carga**
* Por defecto, la carga es de 5 kg. Usa una constante para ello.
* Los métodos que se implementara serán:
  + Getters / Setters necesarios
  + **precioFinal()**:, según el consumo energético se aumenta el precio final y además si tiene una carga mayor de 30 kg, aumentara el precio 50 €, sino es así no se incrementara más el precio.

Crearemos una clase llamada **Television** con las siguientes características:

* Sus atributos propios son **resolución** (en pulgadas) y **sintonizador TDT** (booleano).
* Por defecto, la resolución será de 20 pulgadas y el sintonizador será false.
* Los métodos que se implementara serán:
  + Getters / Setters necesarios
  + **precioFinal()**: según el consumo energético se aumenta el precio final y además si tiene una resolución mayor de 40 pulgadas, se incrementara el precio un 30% y si tiene un sintonizador TDT incorporado, aumentara 50 €.

Ahora crea una clase ejecutable que realice lo siguiente:

* Crea un array de Electrodomesticos de 10 posiciones.
* Asigna a cada posición un objeto de las clases anteriores con los valores que desees. (asigna algunos valores de consumo y color erróneos)
* Ahora, recorre este array y ejecuta los métodos **comprobarConsumoEnergetico(),**  **comprobarColor()** y **precioFinal()** para cada electrodoméstico.
* Deberás mostrar el precio de todas las televisiones, por un lado, el de las lavadoras por otro.

**CINE**

Nos piden hacer un programa orientado a objetos sobre un cine (solo de una sala) tiene un conjunto de asientos (3 filas por 5 columnas).

Del cine nos interesa conocer la película que se está reproduciendo y el precio de la entrada en el cine.

De las películas nos interesa saber el título, duración, edad mínima y director.

Del espectador, nos interesa saber su nombre, edad y el dinero que tiene.

Los asientos son etiquetados por una letra (columna) y un número (fila), como se muestra en la tabla. También deberemos saber si está ocupado o no el asiento.

1A 1B 1C 1D 1E

2A 2B 2C 2D 2E

3A 3B 3C 3D 3E

Realizaremos una pequeña simulación, en el que generaremos entre 5 y 10 espectadores y los sentaremos aleatoriamente de uno en uno (no podemos donde ya este ocupado).

Solo se podrá sentar si tienen el suficiente dinero, hay espacio libre y tiene edad para ver la película, en caso de que el asiento este ocupado le buscamos uno libre.  
Los datos del espectador y la película pueden ser totalmente aleatorios.

La simulación deberá ir mostrando los pasos de cálculo, por ejemplo:

* Llenando la sala para la película “…” con precio X€ para mayores de X edad.
* Sentando al espectador Pedro, con X años y X dinero
* El espectador no tiene suficiente dinero
* Sentando a la espectadora Ana, con X años y X dinero
* Sentada en el asiento 1C
* Sentando a la espectadora Silvia, con X años y X dinero
* La espectadora no tiene suficiente edad
* Sentando al espectador Daniel, con X años y X dinero
* El asiento 1C está ocupado, tratando de buscar otro asiento
* Sentado en el asiento 2B
* …
* Se han gestionado todos los espectadores 😊

**RULETA RUSA**

Se trata de un número de jugadores que con un revolver con una sola bala en el tambor se dispara en la cabeza.

Las clases a hacer son:

* Revolver (con un tambor de 6 huecos):
  + Atributos:
    - posición actual (posición del tambor donde se dispara, puede que esté la bala o no)
    - posición bala (la posición del tambor donde se encuentra la bala)

Estas dos posiciones, se generarán aleatoriamente.

* + Métodos:
    - disparar(): devuelve true si la bala coincide con la posición actual
    - siguienteBala(): cambia a la siguiente posición del tambor
    - informa(): muestra información del revolver (posición actual y donde está la bala)
* Jugador:
  + Atributos
    - id (representa el número del jugador, empieza en 1)
    - nombre
    - nickname (Empezara con el nombre más su ID, “Jugador 1” por ejemplo)
    - vivo (indica si está vivo o no el jugador)
  + Metodos:
    - jugar(Revolver r): el jugador se apunta y se dispara, si la bala se dispara, el jugador muere.
* Juego:
  + Atributos:
    - Jugadores (conjunto de Jugadores)
    - Revolver
  + Funciones:
    - finJuego(): cuando un jugador muere, devuelve true
    - ronda(): cada jugador se apunta y se dispara, se informara del estado de la partida (El jugador se dispara, no ha muerto en esa ronda, etc.)

El número de jugadores será decidido por el usuario, pero debe ser entre 1 y 6. Si no está en este rango, por defecto será 6.

En cada turno uno de los jugadores, dispara el revólver, si este tiene la bala  el jugador muere y el juego termina.

# PER PENSAR COM ENFOCAR

**BARAJA**

Baralla espanyola. Una carta té un número entre 1 i 12 (el 8 i el 9 no els incloem) i un pal (espases, bastos, ors i copes)  
La baralla estarà composta per un conjunt de cartes, 40 exactament.  
Les operacions que podrà realitzar la baralla són:

• barajar: canvia de posició totes les cartes aleatòriament  
• siguienteCarta: retorna la següent carta que està en la baralla, quan no hi hagi més o s'hagi arribat al final, s'indica a l'usuari que no hi ha més cartes.  
• cartasDisponibles: indica el nombre de cartes que encara pot repartir  
• darCartas: donat un nombre de cartes que ens demanin, li retornarem aquest nombre de cartes (pensa que pots retornar). En cas que hi hagi menys cartes que les demanades, no retornarem res però hem d'indicar-li-ho a l'usuari.  
• cartasMonton: vam mostrar aquelles cartes que ja han sortit, si no ha sortit cap indicar-li-ho a l'usuari  
• mostrarBaraja: mostra totes les cartes fins al final. És a dir, si es treu una carta i després es diu al mètode, aquest no mostrés aquesta primera carta.

**PORRA**

Cal realitzar un programa que simuli una porra de la lliga de pàdel.  
Cada jugador/a de la porra, aposta un 1 euro cada jornada i decideix el resultat de dos partits acordats. Els partits són a un set per tant cal endevinar el resultat del set.  
Si ningú encerta en una jornada els resultats, el pot s'acumula. Si algú encerta tots dos partits, s'emporta el pot.  
Totes les persones comencen amb uns diners inicials que serà decidit pel developer. Si un jugador/a no té diners en una jornada no podrà jugar la porra.  
Entre jugadors/as podran repetir resultats, si això passa, es divideix el pot.  
Els resultats de la porra seran generats aleatòriament, tant els de jugador com els dels partits.  
Al final del programa (10 rondes de jocs), s'haurà de mostrar els diners que tenen els jugadors i el nombre de vegades que han guanyat.

Recursivitat

