Sesion 3. Estrucuturas de control iterativas

Empezaremos remarcando de nuevo que C es un lenguaje de programación que requiere de su compilación antes de poder ejecutar el programa. Sin embargo, gracias a Jupyter, ese paso se hace automáticamente. Para poder ejecutar un código en este entorno, basta con seleccionar la celda que contiene el bloque de código y pulsar las teclas **Shift+Enter**.

Sección 1

En el vídeo nos pregunta cómo podríamos mostrar los 1000 primeros números naturales, ejecuta el programa y comprueba que muestra.

In [ ]:
```
#include <stdio.h>

int main(void)
{
    int inicio = 0;
    int fin = 1000;
    int i;
    for (i=inicio; i<fin; i++) {
        printf("%d,", i);
    }
    return 0;
}
```

Sección 2

Para esta sección se nos pide que mostremos sólo los número impares. Utilizando la estructura dada en la explicación, sustituye los interrogantes por lo que creas necesario para que fucione.

In [ ]:
```
#include <stdio.h>

int main(void)
{
    int inicio = ???;
    int fin = 1000;
    int i;
    for (i=inicio; i ??? ???; i=i+???) {
        printf("%d,", i);
    }
    return 0;
}
```

Aun así, parece que hay algo que no cuadra, la última coma debería ser un punto. Modifica el programa anterior para que el último elemento a mostrar vaya acompañado de un punto en lugar de una coma. Para ello, por ejemplo podrías decirle al bucle que haga una iteración menos y una vez termine colocar un **printf** con el valor de **i**. O también podrías poner una condición (recuerdas la estructura **if-else**) dentro del bucle que comprobará si es el último elemento de la lista para ejecutar un **printf** distinto.

Sección 3

While es otro de los elementos que veremos en esta sesión. Para ver como funciona, resolveremos el problema de la "coma en el final" utilizando **while** en lugar de **for**.

In [ ]:
```
#include <stdio.h>

int main(void)
{
    int inicio = 1;
    int fin = 1000;
    int i = inicio;
    while (i < fin-1) {
        printf("%d,", i);
        i = i + 2;
    }
    printf("%d.", i);
    return 0;
}
```

¿se parece a vuestra solución de la sección 2?

Ejercicio: Calculadora

Partiendo de la "Calcooladora v2.0" implementaremos la misma calculadora pero sin utilizar los operadores de multiplicación y división (*, /). Así pues, deberemos implemetar la multiplicación y la división con bucles y los operadores de suma y resta.

Para ayudarte, te damos la estructura del programa:

In [ ]:
```c
#include <stdio.h>

int main(void)
{
    /*  Declaración de variables. */
    float num1 = 3;
    float num2 = 4;
    int op = 3; //0: suma, 1: resta, 2: multiplicación, 3: división.
    float res;

    /* Cuerpo de programa */
    printf("Bienvenido/a a \"Calcooladora v3.0\".\n");
    printf("\n");
    printf("Primera cifra:\t%f\n", num1);
    printf("Segunda cifra:\t%f\n", num2);

    switch (op){
        case (0):
            printf("Operación:\tSuma\n");
            res = num1 + num2;
            break;
        case (1):
            printf("Operación:\tResta\n");
            res = num1 - num2;
            break;
        case (2):
            printf("Operación:\tMultiplicación\n");
            //implementa la operación res = num1 * num2 sin utilizar el
operador de multiplicación.
            break;
        case (3):
            printf("Operación:\tDivisión\n");
            //implementa la operación res = num1 / num2 sin utilizar el
operador de división.
            break;
        default:
            printf("Código de operación no valido.\n");
    }

    printf("Resultado:\t %0.2f\n", res);

    printf("¡Adiós!\n");
    return 0;
}
```

Bonus

Implementa un código en C que calcule el factorial de un número. Recuerda que el factorial de un número natural es el producto de todos los números menores e iguales a él. Por ejemplo: factorial de 5 es 120 (1 x 2 x 3 x 4 x 5 = 120).

In [ ]:
```c
#include <stdio.h>

int main(void)
{
    return 0;
}
```