Sesion 1. Mi primer programa en C

Como ya se ha explicado en los videos, 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. Puedes probarlo en la siguiente celda:

In [ ]:
```
int main(void)
{
    return 0;
}
```

El código anterior no produce ninguna salida por lo que en realidad, para saber si lo has hecho bien te has de fijar en el numero en la parte derecha (In [X]). si cambia, lo estás haciendo bien.

Ahora vamos a analizar el código anterior de donde podemos obtener la estructura necesaria para un programa en C. La primera línea contiene una palabra reservada por el lenguaje (`main`) y que indica el primer paso que será ejecutado. La porción de código que pertenece a main está delimitada por el uso de llaves `{}`. Por otro lado, la línea formada por `return 0;` indica la finalización del programa. Por lo tanto, el programa anterior solamente se inicia y acaba, por eso no se observa nada. ;)

En el siguiente bloque se muestra un código "más complejo" y en el que se observan dos líneas nuevas. Empecemos por la segunda. printf es una función que nos permite imprimir resultados por pantalla. Si ejecutais el siguiente bloque de código vereis el resultado del programa en la parte inferior del recuadro. Por otro lado, tenemos la primera línea que solamente sirve para que la función `printf` pueda ser ejecutada.

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

int main(void)
{
    printf("Hello world!!\n");
    return 0;
}
```

¡Bien hecho!
Ahora, vamos a explicar pequeños errores muy comunes a la hora de escribir código en C. En el bloque anterior, utilizamos la línea `#include<stdio.h>` para permitir el uso de la función `printf`. Probemos a ejecutar el código sin esta línea (siguiente bloque).

In [ ]:
```
int main(void)
{
    printf("Hello world!!\n");
    return 0;
}
```

Aunque el resultado obtenido es el correcto, los mensajes que aparecen sobre fondo rosaceo nos indican que hay algún error o warning. Este warning nos indica que línea podría no ser ejecutada y ademas nos sugiere cómo corregirlo.
Nota de los profesores: Aunque el compilador parece amistoso, la mayoría de las veces no lo es.

Ejercicio
Modifica el ejemplo anterior para eliminar el warning.

Otro aspecto a tener en cuenta es que todas las líneas del código en C que no lleven asociados paréntesis, deben acabar con el símbolo `;`. Este símbolo indica que es el final de una orden. Veamos que ocurre si se nos olvida poner este carácter.

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

int main(void)
{
    printf("Hello world!!\n")
    return 0;
}
```

En este caso, vemos como el código no se ejecuta ya que ahora aparece un error en lugar de un warning. Más o menos intenta decir que y donde se nos ha olvidado.

Ejercicio
Modifica el ejemplo anterior para eliminar el error y ejecuta el programa.

En el siguiente código se muestra todo lo que aprenderemos en esta sesión.
¿Eres capaz de entenderlo?
Si no, tranquilo, ahora lo explicamos ;)
No desperdicies la oportunidad de ejecutarlo y modificarlo, tal vez te sea útil para el último ejercicio...

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

int main(void)
{
    int a,b;
    int c;
    a = 3;
    b = 6;
    c = a+b;
    printf("El resultado de %d+%d es %d!!\n",a,b,c);
    return 0;
}
```

Tipos de datos y marcas de formato

En el siguiente ejemplo mostramos cómo: declarar, inicializar y representar datos según su tipo.
Modifícalo para que muestre en la salida la fecha de hoy y tu nombre.

In [ ]:
```c
/* Bibliotecas que utiliza el programa */
#include <stdio.h>

/* Punto de inicio del programa */
int main(void) {

    /* Declaraciones e inicializaciones */
    char *name = "Sergio";
    float height = 1.91;
    int day = 1;
    int month = 1;
    int year = 2017;

    /* Cuerpo del programa */
    printf("Soy %s y mido %0.2f metros.\nHoy es %d del %d de %d y digo:\
t¡Hola!\n",
            name, height, day, month, year);

    /* Acaba la ejecución del programa */
    return 0;
}
```

¿Qué puede significar el **%0.2f** en la marca de formato?
Prueba a quitarlo y dejar únicamente **%f**...

Ejercicio: Calculadora

Tal y como se explica en el material de la sesión 1, implementa:
un programa en C que dados dos números calcule su: **suma, resta, producto y división**.
Los dos número deberán almacenarse en variables.
Los resultados de las operaciones deberán guardarse en variables.
Tras la ejecución del programa se mostrará para cada operación su resultado.

Para ayudarte, te damos la estructura del programa:

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

int main(void)
{
    /*  Declaración de variables. */
    float num1 = 5;
    float num2 = 2;

    /*  Elimina los comentarios y sustituye los
        interrogantes por el tipo de datos correspondiente. */
    // int sum;
    // ??? res;
    // ??? prod;
    // ??? div;

    /* Cuerpo de programa */
    printf("Bienvenido/a a \"Calcooladora v1.0\".\n");
    printf("\n");
    printf("Primera cifra:\t%f\n", num1);
    printf("Segunda cifra:\t%f\n", num2);
    // Añade un salto de linea más.

    printf("...calculando resultados...\n");
    // Falta otro salto de línea...

    // Calcula suma (este te lo damos hecho)
    //sum = num1 + num2;
    //printf("Suma:\t\t%d\n", sum);

    // Calcula resta (no te olvides de sustituir los interrogantes...)
    //res = ???
    //printf("Resta:\t\t%d\n", ???);

    // Calcula producto

    // Calcula división (cuidado con los tipos de las variables)

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