Sesion 4. Conjuntos de datos y funciones

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 siguiente ejemplo vemos como declarar e inicializar un vector. Además, tras mostrarlo, modificamos su contenido accediendo a cada una de sus posiciones.

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

int main(void)
{
    int vector[5] = {2,4,5,1,7};
    int i;
    for (i=0; i<5; i++) {
        printf("%d ", vector[i]);
    }
    printf("\n");
    for (i=0; i<5; i++) {
        vector[i] = (i+1)*10;
    }
    printf("\n");
    for (i=0; i<5; i++) {
        printf("%d ", vector[i]);
    }

    return 0;
}
```

Modifica el código anterior para que tras mostrar el vector modificado, muestre la suma acumulada de todos sus elementos. Por ejemplo: para el vector {10,20,30,40,50} el resultado sería 150.

Sección 2

En esta sección trabajaremos con cadenas. Fíjate primero, en el código, como hemos incluido la biblioteca para poder utilizar sus funciones. Ejecuta el siguiente código y trata de entender lo que hace:

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

int main(void)
{
    char str1[7] = "cuarta";
    char str2[2] = " ";
    char str3[8] = "y mitad";
    char str4[15];

    strcpy(str4, str1); //copiamos la cadena "str1" en "str4".
    strcat(str4, str2); //concatenamos "str4" y "str2".
    strcat(str4, str3);

    printf("%s", str4);

    return 0;
}
```

Esto también se podría haber hecho inicializando "str4" y concatenando el resto de cadenas:

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

int main(void)
{
    char str1[7] = "cuarta";
    char str2[2] = " ";
    char str3[8] = "y mitad";
    char str4[15] = "";

    strcat(str4, str1);
    strcat(str4, str2); //concatenamos "str4" y "str2".
    strcat(str4, str3);

    printf("%s", str4);

    return 0;
}
```

Y finalmente, veremos que también podemos tener un vector de cadenas (piénsalo bien, en el fondo es una matriz de carácteres). En este ejemplo hemos declarado un vector de cadenas. Como las cadenas puedes tener distinta longitud, las declararemos de 10 carácteres (¡cuidado estamos desperdiciando memoria!).

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

int main(void)
{
    char strs[5][10] = {{"primero"}, {"segundo"}, {"tercero"}, {"cuarto"
}, {"quinto"}};
    int i;
    for(i = 0; i < 5; i++){
        printf("%s ", strs[i]);
    }

    return 0;
}
```

Además, como este vector tiene dos dimensiones (matriz), podemos recorrer carácter a carácter las cadenas:

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

int main(void)
{
    char strs[5][10] = {{"primero"}, {"segundo"}, {"tercero"}, {"cuarto"
}, {"quinto"}};
    int i, j, len;
    for(i = 0; i < 5; i++){
        len = strlen(strs[i]); //calcula el número de carácteres de la c
adena.
        for (j = 0; j < len; j++) {
            printf("%c ", strs[i][j]);
        }
    }

    return 0;
}
```

Esto se pone interesante... Para cada cadena, calculamos su número de carácteres (hasta el "\0" que indica el final de la palabra pero no se ve) y los recorremos. Con lo visto hasta ahora, ¿puedes sustituir todas las vocales del vector por "x"?

Sección 3

En el siguiente código hemos definido la función "suma" a la que le damos dos números (parámetros) y nos devuelve un "int".

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

int suma(int num1, int num2) {
    return num1 + num2;
}

int main(void)
{
    int a = 5, b = 6, res;
    res = suma(a,b);
    printf("%d", res);
    return 0;
}
```

Implementa la función "getMax" que dados dos números devuelva el mayor de ambos.

Ejercicio: Calculadora

Implementa una calculadora que además de números opere con cadenas. Dependiendo del modo de funcionamiento (números o cadenas), la calculadora llamará a unas funciones u otras (función suma, función resta...).

Para el modo “números” la salida será parecida a la imagen de la izquierda. Ya que estamos en el tema de las funciones, cada operación debe implementarse en una función distinta.

Para el modo “cadenas” (imagen de la derecha), implementaremos utilizando la biblioteca , una función que nos devuelva la suma de los carácteres de las cadenas.

Para ayudarte, te damos la estructura del programa:

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

float suma (float a, float b) {
    return a + b;
}

float modoNum(float n1, float n2, int op){
        float res;
        switch (op){
        case (0):
            printf("Operación:\tSuma\n");
            res = suma(n1,n2);
            break;
        case (1):
            printf("Operación:\tResta\n");
            //res = num1 - num2;
            break;
        case (2):
            printf("Operación:\tMultiplicación\n");
            //
            break;
        case (3):
            printf("Operación:\tDivisión\n");
            //
            break;
        default:
            printf("Código de operación no valido.\n");
    }

    return res;
}

int modoChar(char str1[], char str2[]) {
    //...
}

int main(void)
{
    /*  Declaración de variables. */
    int modo = 1; //0: números, 1: cadenas.
    int op = 0; //operación
    /*
    * operación en modo números: 0: suma, 1: resta, 2: multiplicación, 3
: división.
    * operación en modo cadenas: no tiene efecto, sólo tenemos una opera
ción.
    */

    float res;

    if (modo == 0) {
        float num1 = 3;
        float num2 = 4;
        res = modoNum(num1, num2, op);
    } else if (modo == 1) {
        char str1[10] = "cuarta";
        char str2[10] = "sesión";
        res = modoChar(str1, str2);
    }

    /* Cuerpo de programa */
    printf("Bienvenido/a a \"Calcooladora v4.0\".\n\n");

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

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

¿Has reutilizado la función "suma" para el modo "cadenas"?