EJERCICIOS – UT 0

Los ejercicios del tema 0 se desarrollarán en C.

Es recomendable generar una carpeta llamada **DAM2_PSYP_EJER0001_APE1_APE2_NOM** y dentro de ésta los ejercicios DAM1_PSYP_EJER0001_XX_APE1_APE2_NOM.c, siendo XX números de 01 hasta 10.

Ejercicios

1. Hacer un programa al cual introduzca un número y me diga si es mayor que 5 o menor
2. Crea un programa que pida dos números al usuario y muestre sus posiciones de memoria.
   Para imprimir las posiciones de memoria hay que usar %p.
   Muestra posteriormente cuanto tamaño ocupa en memoria un entero usando la función *sizeof(variable/tipovariable)* que devuelve el número de bytes que ocupa una variable en memoria.
   Teniendo en cuenta que cada posición de memoria abarca un byte, intenta explicar si las variables están seguidas en memoria o no.
3. Hacer un programa al cual introduzca tres números por teclado y me los devuelva ordenados
4. Hacer un programa que me solicite dos números y me ofrezca un menú, en el cual las opciones sean las siguientes:
   a. Sumar
   b. Restar
   c. Multiplicar
   d. Dividir
   e. Salir

   En función de la opción seleccionada se obtendrá el resultado conveniente.

   Tened en cuenta que el buffer almacena el enter que pulsamos tras introducir un valor en *scanf*, por tanto, si solicitamos un carácter después de haber pedido otro valor, se rellenará solo son ese enter.
   Podéis utilizar la función *getchar()*; o introducir un espacio donde corresponda en el *scanf* para que se vuelque ese carácter del buffer *scanf(" %c", &opcion);.*

5. Modifica el programa anterior de manera que hasta que el usuario no seleccione la opción de salir se repita el comportamiento.
6. Modulariza el programa anterior, de manera que cada opción sea una función.
7. Escribir un programa que me pida un número del 1 al 20, si el usuario introduce mal este valor debe volver a solicitarlo.

El programa ha de mostrar la tabla de multiplicar del número.

8. Escribe un programa que pida al usuario 10 caracteres.
   Una vez introducidos, el programa ha de comprobar si el usuario ha introducido el carácter “k” y en el caso de que sea así ha de indicarlo de manera eficiente.
   Es decir, si el carácter k es el segundo introducido por el usuario, el programa no necesitará analizar los números introducidos por el usuario posteriormente.
   Para usar booleanos es necesario incluir una biblioteca: *#include<stdbool.h>*
9. Crea un programa que pida al usuario una palabra y la invierta .
   Se puede usar la función *strlen(cadenaDeCaracteres)* para saber el número de caracteres que tiene la cadena a analizar.
   Se encuentra dentro de la biblioteca *<string.h>*.
10. Crea un programa que pida al usuario un lado de la matriz (cuyo valor máximo ha de ser 10 y el mínimo 2).
    El programa ha de generar dos matrices con números aleatorios (entre 0 y 9) de las dimensiones indicadas por el usuario y mostrarlos ordenadamente por pantalla.
    El programa multiplicará estos arrays y mostrará el resultado.
    Como no tenemos conocimiento en punteros, no es necesario modularizar el programa.

LA FUNCIÓN rand()

En C, para obtener números aleatorios, tenemos la función **rand()**. Esta función, cada vez que la llamamos, nos devuelve un número entero aleatorio entre **0** y el **RAND_MAX** (un número enorme, como de 2 mil millones).

El primer problema que se nos presenta es que no solemos querer un número aleatorio en ese rango, sería un dado muy curioso el que tenga tantas caras. Podemos querer, por ejemplo, un número aleatorio entre **0** y **10**. O de forma más general, entre **0** y **N**. La cuenta que debemos echar para eso es esta

```
#include <stdlib.h>
...
numero = rand() % 11;
numero = rand() % (N+1);
```

La operación módulo (**%**) nos da el resto de dividir **rand()** entre **11**. Este resto puede ir de **0** a **10**. De la misma forma, el módulo de **rand()** entre **N+1** va de **0** a **N**.

¿Y si queremos un rango que no empiece en **0**?. Por ejemplo, queremos un rango entre **20** y **30** (de forma más general, entre **M** y **N** con **N** mayor que **M**). Pues es fácil, obtenemos un número entre **0** y **10** y le sumamos **20** (un número entre **0** y **N-**M y le sumamos **M**)

```
numero = rand () % 11 + 20;  // Este está entreo 20 y 30
numero = rand () % (N-M+1) + M;   // Este está entre M y N
```

Rand siempre devuelve el mismo valor