OBJETIVO:

Aprender de forma práctica la utilización del depurador con las funciones en C.

1. Escribe un programa que reciba por teclado un número entero comprendido entre 0 y 10.000, calcule la suma de sus cifras y lo muestre por pantalla. El programa seguirá solicitando números hasta que el usuario desee terminar (dando las opciones s/n, 's' para continuar, 'n' para finalizar).

Para ello, se utilizarán al menos dos funciones:

- La función *int validar (int menor, int mayor);* que pida un número comprendido entre menor y mayor. Si el número introducido es incorrecto, solicitará otro. Si es correcto, lo devolverá como resultado.
- Y la siguiente función recursiva:

```
int sumaCifras (int num) {
        int resul = 0;
        if (num > 9)
              resul = num%10+sumaCifras (num/10);
        else resul = num;
        return resul;
}
```

Copia a continuación el programa desarrollado.

```
#include <stdio.h>
#include <stdlib.h>
int validar(int menor, int mayor);
int sumaCifras(int num);

int validar (int menor, int mayor)
{
   int num;
   do
   {
   printf("Introduce un numero entre %d y %d:", menor, mayor);
   scanf("%d", &num);
   if (num<menor || num>mayor)

      printf("No es correcto\n");

   }while(num<menor || num>mayor);

   return num;
}
int sumaCifras (int num)
 {
int resul = 0;
if (num > 9)
resul = num%10+sumaCifras (num/10);
else resul = num;
```

```
return resul;
 }

int main()
{
   char s;
   int num, mayor, menor;
   num=validar(0,10000);
   printf("La suma de sus cifras es %d:\n", sumaCifras(num));
   printf("Quieres volver a ntroducir un numero? 's' para si 'n' para no:");
   scanf("%s", &s);
   while(s!='n')
   {
   num=validar(0,10000);
   printf("La suma de sus cifras es %d:\n", sumaCifras(num));
   printf("Quieres volver a ntroducir un numero? 's' para si 'n' para no:");
   scanf("%s", &s);
   }

   return 0;
}
```

Verifica su correcto funcionamiento utilizando el depurador.

a. Pon un punto de ruptura (*breakpoint*) en la línea en la que se invoca la función *sumaCifras*. A continuación, ejecuta el programa utilizando el *debug* e introduciendo el valor 5874. Además de la ventana *Watches* (*Debuggin Windows*), añade la ventana *Call stack*.
b. Ve ejecutando paso a paso el programa (utilizando Step into/Mayús F7 o el botón ).
c. Continúa la ejecución hasta que *num* tenga una sola cifra. Haz capturas de pantalla de las ventanas Watches y Call stack en ese momento.

Watches	Call stack

Cuando se ejecuta una función recursiva, se van a guardar los siguientes datos en una parte de la memoria del sistema (la pila del sistema o *Stack*):

- Una copia de la función *main*, con sus variables locales, y el punto en el que se debe continuar la ejecución (llamada a la función recursiva).
- Por cada llamada a una función, también se guarda en memoria una copia de la función, de sus variables locales, y del punto en el que se hace la llamada.
- Cada vez que finaliza la ejecución de una llamada a la función, dicha función se borra de la pila del sistema

d. Teniendo en cuenta la explicación anterior, analiza las capturas que has hecho en el apartado anterior.

e. Sigue ejecutando el programa hasta que en la ventana *Call stack* queden solo la función *main* y una llamada a la función *sumaCifras*. Haz dos nuevas capturas de pantalla.

Watches	Call stack

f. Continúa ejecutando el programa hasta su finalización.

2. Escribe un programa que reciba por teclado un número entero comprendido entre 0 y 10.000, y verifique si cumple la siguiente condición: la suma de sus cifras es múltiplo de la cifra de las unidades. El programa seguirá solicitando números hasta que el usuario desee terminar (dando las opciones s/n, 's' para continuar, 'n' para finalizar).

Para ello, se utilizarán al menos dos funciones:

- La función *int validar (int menor, int mayor);* que pida un número comprendido entre menor y mayor. Si el número introducido es incorrecto, solicitará otro. Si es correcto, lo devolverá como resultado.
- La función *void analizar (int n, int *sumCifras, int *unidades);* que a partir de n, valor entero, devuelve mediante el uso de los punteros *sumCifras* y *unidades,* la suma de las cifras de n y el valor de las unidades.

Copia a continuación el programa desarrollado.

```
#include <stdio.h>
#include <stdlib.h>
int validar(int menor, int mayor);
int sumaCifras(int num);
void analizar (int num, int *sumCifras, int *unidades);

int main()
{
    char s;
    int mayor, menor, num, unidades, sumCifras;

    num=validar(0,10000);

    printf("La suma de sus cifras es %d\n", sumaCifras(num));

    analizar(num, &sumCifras, &unidades) ;

    printf("\n\nQuieres volver a introducir un numero? 's' para si 'n' para no:");
  scanf("%s", &s);
  while(s!='n')
 {
   num=validar(0,10000);
   printf("La suma de sus cifras es %d\n", sumaCifras(num));
   analizar(num, &sumCifras, &unidades);
   printf("\n\nQuieres volver a introducir un numero? 's' para si 'n' para no:");
   scanf("%s", &s);
  }

return 0;
}
```

```
int validar(int menor, int mayor)
{
    int num;
    do
    {
        printf("Introduce un numero entre %d y %d: ", menor, mayor);
        scanf("%d", &num);
        if (num<menor || num>mayor)

        printf("El numero no es correcto");

    } while (num<menor || num>mayor);

    return num;

}
int sumaCifras(int num)
{
    int resul = 0;
    if (num> 9)
        resul = num%10+sumaCifras (num/10);
    else resul = num;
    return resul;

}
void analizar (int num, int *sumCifras, int *unidades)
{
    *unidades = num%10;
    *sumCifras= sumaCifras(num);

    if( *sumCifras % *unidades == 0)

    printf("La suma de las cifras es multiplo de la cifra de las unidades.");

    else

    printf("La suma de sus cifras no es multiplo de la cifra de las unidades.");

}
```

Verifica su correcto funcionamiento utilizando el depurador.

a. Pon un punto de ruptura (*breakpoint*) en la línea en la que se invoca la función *analizar*. A continuación, ejecuta el programa utilizando el debug e introduciendo el valor **444**. Además de la ventana *Watches* (*Debuggin Windows*), añade la ventana *Call stack*.

b. Ve ejecutando paso a paso el programa (utilizando Step into/Mayús F7 o el botón ).

c. Continúa la ejecución hasta que *n* tenga una sola cifra. Haz capturas de pantalla de las ventanas Watches y Call stack en ese momento.

Watches	Call stack

d. Analiza las capturas que has hecho en el apartado anterior.

e. Continúa ejecutando el programa hasta su finalización.

3. Escribe un programa que reciba por teclado un número entero, n, comprendido entre 0 y 100, calcule el valor del n-simo término de la serie de Fibonacci y lo muestre por pantalla. El programa seguirá solicitando números hasta que el usuario desee terminar (dando las opciones s/n, 's' para continuar, 'n' para finalizar).

Para ello, se utilizarán al menos dos funciones:

- Una función de prototipo *int validar (int menor, int mayor);* que pida un número comprendido entre menor y mayor. Si el número introducido es incorrecto, solicitará otro. Si es correcto, lo devolverá como resultado.
- Y la siguiente función (de forma iterativa):

```
int Fibonacci  (int n) {
    //…Completarla: A(0) =0; A(1) =1; A(N) =A(N-1)+ A(N-2);
}
```

Verifica el correcto funcionamiento de las funciones. Si es necesario, utiliza el *debug* para detectar los posibles fallos que se puedan producir.

Copia a continuación el programa desarrollado.

```
#include <stdio.h>
#include <stdlib.h>
int validar(int menor, int mayor);
int Fibonacci(int num);

int main()
{
   char s;
   int num, menor, mayor;

   num=validar(0,100);
   printf("La posicion introducida corresponde al numero %d en la serie de Fibonacci ", Fibonacci(num));
   printf("\n\nQuieres volver a introducir un numero? 's' para si 'n' para no:");
   scanf("%s", &s);
   while(s!='n')
   {
      num=validar(0,100);
   printf("La posicion introducida corresponde al numero %d en la serie de Fibonacci ", Fibonacci(num));
      printf("\n\nQuieres volver a introducir un numero? 's' para si 'n' para no:");
      scanf("%s", &s);
   }

   return 0;
}



int validar(int menor, int mayor)
```

```
{
   int num;
   do
   {
printf("Introduce un numero entre %d y %d:", menor, mayor);
scanf("%d", &num);
   if (num<menor || num>mayor)

printf("No es correcto\n");

}while(num<menor || num>mayor);

   return num;
}

int Fibonacci (int num)
{
   if (num==1 || num ==2)
      return 1;
   else
      return (Fibonacci(num-1)+Fibonacci(num-2));
}
```