CAPITULO 4: ENTRADA Y SALIDA DE DATOS.

1. INTRODUCCIÓN.

Hemos visto que el lenguaje C va acompañado de una colección de funciones de biblioteca que incluye un cierto número de funciones de entrada/salida. En este capitulo, se verán algunas funciones que nos permiten la transferencia de datos entre el ordenador y los dispositivos de entrada/salida estándar, que son el teclado y el monitor.

Para facilitar el uso de éstas funciones (o cualquier función de biblioteca), se incluyen unos archivos de cabecera que proporcionan la información necesaria para el uso de un determinado grupo de ellas. Esta información consiste básicamente en la definición de algunas constante, el prototipo de las funciones.

El conjunto de las funciones de entrada /salida están definidas en el fichero **stdio.h,** pero en algunos compiladores se pude tener algunas funciones de entrada/salida definidas en **conio.h**. Cuando se emplea alguna función de entrada/salida, se debe incluir el fichero al comienzo del programa del programa mediante la directiva **#include <stdio.h>** y el compilador inserta el código fuente al comienzo del archivo.

2. ENTRADA/SALIDA SIN FORMATO.

Para leer un **carácter** desde el teclado, o imprimirlo en el monitor, se emplean las siguientes funciones.

- La función **getche()** espera hasta que se pulse una tecla y devuelve su valor; la tecla pulsada aparece en pantalla.
- La función **getch()** opera como la anterior, pero sin eco, es decir el carácter pulsado no aparece en la pantalla.
- La función **getchar()** lee un carácter del teclado (con eco) y devuelve su valor cuando se pulsa la tecla **INTER**.
- La función **putchar ( )** imprime un carácter en la pantalla.

```
char ch;
ch = getchar();     Lee un carácter y lo asigna a ch.
putchar(ch);        Escribe en pantalla el contenido de ch.
puchar('B');        Escribe el carácter B en pantalla.
puchar( 98);        Igual que antes,98 es el código ASCII de B
```

Para leer o escribir una cadena de caracteres, se usan las siguientes funciones.

- La función **gets( )** lee una cadena de caracteres introducida por el teclado hasta

pulsar la tecla **INTER.** La cadena de caracteres se almacena en un array de caracteres con la dirección apuntada por el argumento de la función y en lugar de nueva línea (**\n**), en su lugar se pone el carácter null (**\n**).

- La función **puts ( )** escribe una cadena de caracteres en la pantalla y traduce el carácter null (**\0**) por un carácter de nueva línea (**\n**).

```
Char *ch, linea[80];    Declaración de un puntero a carácter
........                 y un array de 80 caracteres.

gets(ch);               Lee una cadena que termina con INTER.
gets(linea);            Misma operación que antes.
puts(ch);               Imprime el contenido apuntado por ch
puts(linea);            Igual que antes, el puntero es línea.
puts("UNO");            Escribe La cadena entre las comillas.
```

3. ENTRADA/SALIDA CON FORMATO.

La función printf().

Para visualizar datos por la pantalla, se dispone de la función **printf()** que permite formatear la salida. Esta función puede visualizar una combinación de valores numéricos, caracteres y cadenas de caracteres.

Sintaxis: printf(cadena_de_control, arg1, arg2, ..., argn);

La **cadena_de_control** contiene la información sobre el formato de las salidas de los argumentos; **arg1, arg2, ... etc,** y los argumentos pueden ser constantes, variables, arrays, referencias a funciones, o cualquier expresión C. La **cadena_de_control** está compuesta por grupos de caracteres, donde cada grupo encabezado por el símbolo (**%**) representa el formato del dato de salida. En la cadena de control, se puede intercalar texto si formato para comentar las salidas.

Debe haber igual número de comandos de formato como argumentos. Si hay más argumentos que formatos se descartan los que sobran, pero si hay menos, la salida es indefinida.

Los comandos de formato pueden tener unos modificadores que especifican la longitud de del campo, el número de dígitos decimales, y un indicador de justificación a la izquierda. Estos modificadores se colocan entre el simbolo (**%**) y el comando de formato.

Tabla de los códigos de los formatos posibles.

Formato	Visualización del dato.
%c	Carácter
%s	una cadena de caracteres.
%d	Entero decimal con signo
%o	Entero en octal (sin 0)
%u	Entero decimal sin signo
%x	Entero en hexadecimal (sin 0X)
%f	Coma flotante.
%e	Coma flotante en notación científica.
%g	Coma flotante sin cero ni coma si es el caso.
%p	Dirección del puntero.

Formato	Modificación del dato
%10d	imprime un entero en un campo de 10 dígitos.
%5.2f	imprime un float en un campo de 5 dígitos donde 2 de ellos son decimales.
%06d	rellenara con ceros, en vez de espacios en blanco, los números menores de 6 dígitos.
%-6d	justifica a la izquierda el número en un campo de 6 dígitos.
%5-7s	imprime una cadena de al menos 5 caracteres y no más de 7. Se truncará al final si tiene más de 7 y se rellenara con espacios en blanco si tiene menos de 5

```
char ch = 'A';
int m = 4;
float x = 14.250;

printf("ch=%c, m=%d", ch, m);
printf("ch=%d, x=%f", ch, x);
printf("ch=%x,\n m=%d ", ch, m);

printf("x=%4.1f, m=%05d", x, m);
printf("La suma x+m=%f",(x+m));
printf("La tecla es %c\n",getch());
```

```
El programa produce
las siguientes salidas.


ch = A , m= 4
ch = 65, real = 14.250
ch = 41,
m = 4  (N.B salto de linea).
x = 14.2, m = 00004
Argumento como expresión(x+m)
El argumento es una función.
```

La función scanf().

La función **scanf()** lee los datos desde el teclado y la sintaxis es parecida a la función anterior.

Sintaxis: scanf(cadena_de_control, arg1, arg2, ..., argn);

La **cadena_de_control** contiene la información sobre el formato de las entradas y son idénticos a los formatos vistos en la función **printf().** Pero, los argumentos; **arg1, arg2, etc,** son direcciones de las variables y para ellos se debe utilizar el operador de las direcciones (**&**). Cuando se trata de capturar un array de datos, lo argumentos no debe ir precedido por el ampersand (**&**), ya que el propio nombre del array indica la dirección del mismo.

`Int n;`
`Float x;`
`Char cadena[20];`
`........`
`scanf("%f %d",&x, &n);`
 	
`scanf("%s ", cadena);`
	
	`Dos grupos; El primero es float (%f)`
	`y el segundo es un entero (%d).`
	`Las variables se almacenan en las`
	`direcciones(&x y &n).`
	
	`La función lee una cadena de caracteres.`
	`El argumento no lleva el operador &.`

La lectura de un dato termina cuando encuentra un espacio en blanco, un tabulador o un carácter de nueva línea y estos caracteres se usan como separadores de campo.

```
   scanf("%s %d %f", cadena, &n, &x);

Los datos se pueden introducir de las siguientes maneras.
```

`Mensaje`
`120`
`0.25`
	`Mensaje 120 0.25`	`Mensaje  120`
		`0.25`

Las cadenas de caracteres igual que el resto de los tipos de datos acaban con un espacio en blanco, por lo tanto una cadena de caracteres que incluye caracteres de espaciado no se puede introducir de esta forma.

El carácter de conversión (**% s**) es reemplazada por una secuencia de caracteres enceradas entre corchetes designada como (**[. . .]**). Los caracteres de espaciado pueden ir dentro de los corchetes. En el proceso de lectura de los datos, se leerán únicamente los caracteres entre los corchetes.

Existe una alternativa de interpretar los caracteres entre los corchetes de forma opuesta mediante el operador (^) ; es decir, se leerán los caracteres mientras no coincida con los caracteres incluidos en los corchetes. Veamos el siguiente ejemplo

`Char *cad;`
 	
`Scanf("%s ", cad);`
`Scanf("%[ ABCD]",cad);`
 	
`Scanf("%[^\nABCD]",cad);`
 	
`Scanf("[^\n]",cad);`
	
	`No se puede leer un espacio en blanco.`
	`Lee solo los espacios en blanco y`
	`las letras especificadas( ,A,B,C,D).`
	`Lee todos los caracteres salvo (\n)`
	`y los caracteres A,B,C,D)`
	`Lee todos los caracteres salvo (\n).`

Igual que la función printf(), la función **scanf()** puede limitar el campo de los datos en la entrada.

`scanf("%3d %3d %3d",&a,&b,&c);`	`Datos introducidos.`
	`1 2 3 a=1, b=2, c=3`
	`123 456 789 a=123, b=456, c=789`
	`123456789 a=123, b=456, c=789`
	`1234 5678 9 a=123, b=4, c=567`