Programación de dispositivos electrónicos

Unidad 2

Estructuras de control - Funciones de Entrada y Salida - Lenguaje C

Héctor Albano Diego Codevilla

Proposición if - else

Esta proposición sirve para ejecutar ciertas sentencias de programa ,si una expresión resulta verdadera, u otro grupo de sentencias, si la expresión resulta falsa. Luego se continúa con el programa.

Su sintaxis es:

```
if (expresión)
{
        enunciado1;
}
else
{
        enunciado2;
}
enunciado3;
```

También podría solo ejecutarse sentencias cuando la expresión es verdadera, y continuar con en programa en caso de ser falsa.

Su sintaxis será:

```
if (expresión)
{
        enunciado1;
}
Enunciado3;
```

Si los enunciados tuvieran una sola sentencia se podría abreviar y no colocar las llaves

```
if (expresión)
        enunciado1;
else
        enunciado2;
enunciado3;
```

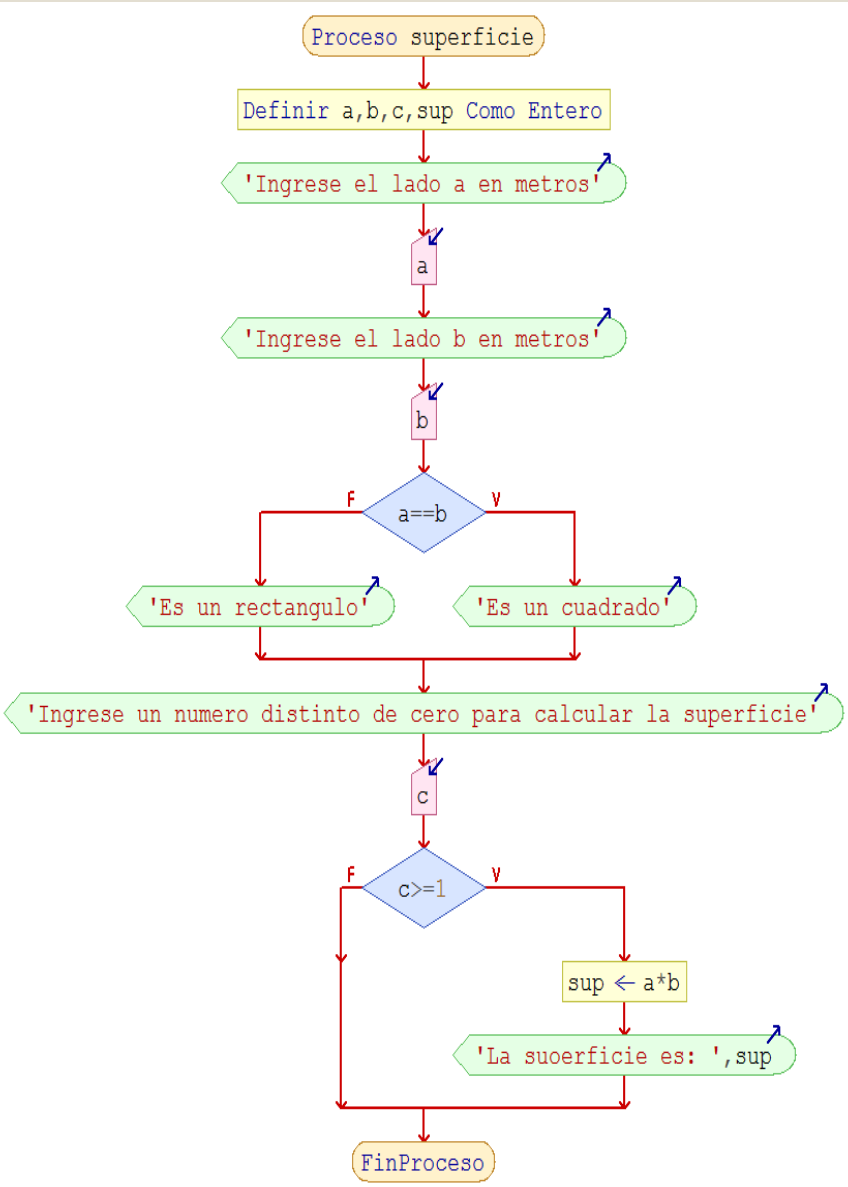
O también:

```
if (expresión)
        enunciado1;
enunciado3;
```

Nota: es muy importante no olvidar las llaves cuando las sentencias a ejecutar tanto en el caso de ser verdadera como falsas son múltiples.

```c
/*Ejemplo de if if else*/
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

int main(void)
{
    int a,b,c,sup;
    printf("Ingrese lado a en metros\n");
    scanf("%d",&a);
    printf("Ingrese lado b en metros\n");
    scanf("%d",&b);
    if(a==b)
        printf("Es un cuadrado");
    else
        printf("Es un rectangulo");
    printf("\nSi desea calcular la superficie ingrese un numero distinto de cero\n");
    scanf("%d",&c);
    if(c>=1)
    {
        sup=a*b;
        printf("La superficie es de %d metros cuadrados",sup);
    }
    printf("\nPresione una tecla para salir");
    getch();
    return 0;
}
```



Proposición while

Esta proposición sirve para ejecutar de forma repetida durante un número finito de veces un conjunto de instrucciones. La condición para que se sigan ejecutando es que la expresión evaluada sea cierta.

Su sintaxis es:

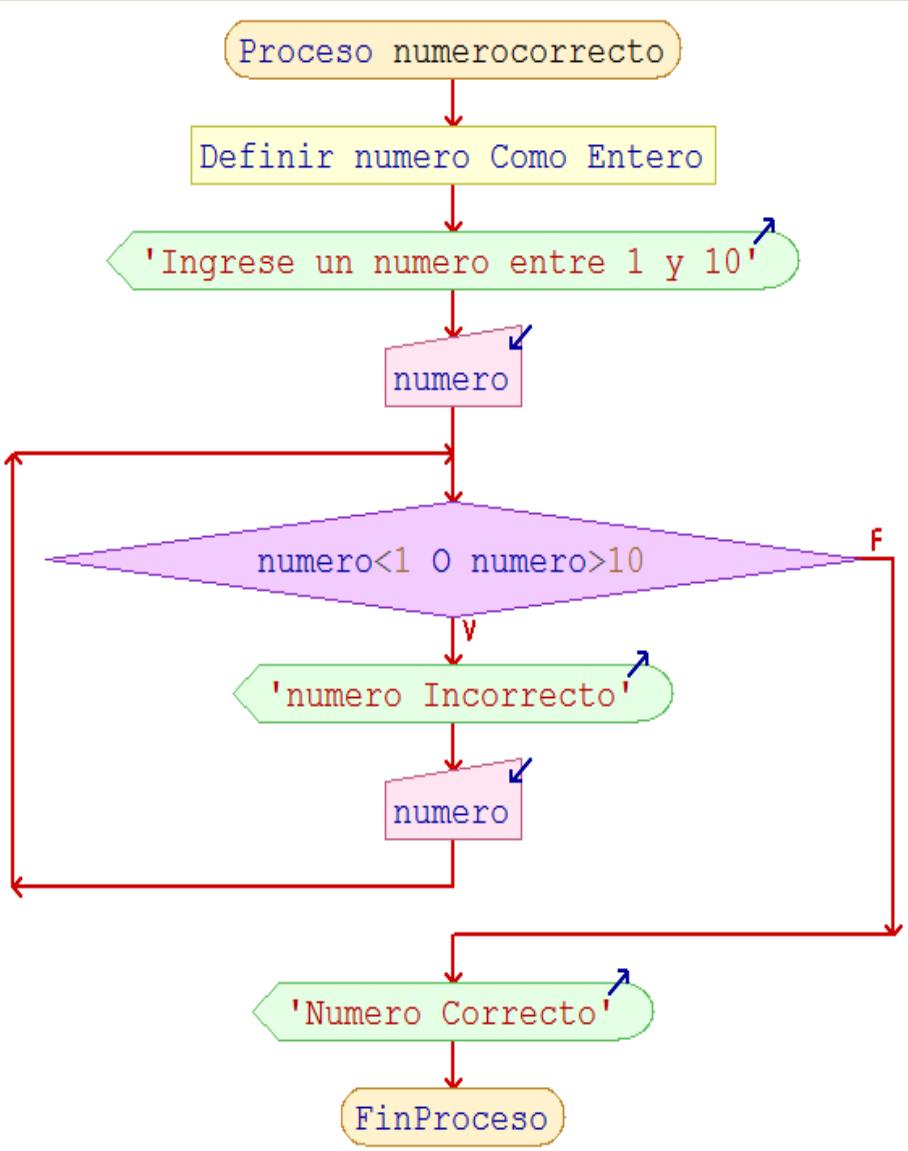
```
while(expresión)
{
        enunciado1;
}
enunciado3;
```

Dentro del enunciado1 debe existir al menos una instrucción que haga que la condición sea falsa, de no ser así nunca saldrá de la proposición while.

1- se evalúa la expresión de la condición
2- si es falsa se termina y se ejecuta el primer enunciado fuera del while (enunciado3)
3- si es cierto se ejecuta el enunciado del while y se vuelve a 1

```c
#include<stdio.h>

/* Ingresar un numero solo si es correcto */
int main() {
        int numero;
        printf("Ingrese un numero entre 1 y 10\n");
        scanf("%i",&numero);
        while (numero<1 || numero>10)
                {
                printf("numero Incorrecto\n");
                scanf("%i",&numero);
                }
        printf("Numero Correcto\n");
        return 0;
}
```



Proposición do....while

Es muy similar a la anterior con la diferencia que las instrucciones al menos se ejecutan una vez, ya que primero se ejecutan los enunciados y luego se evalúa la expresión.

Su sintaxis es:

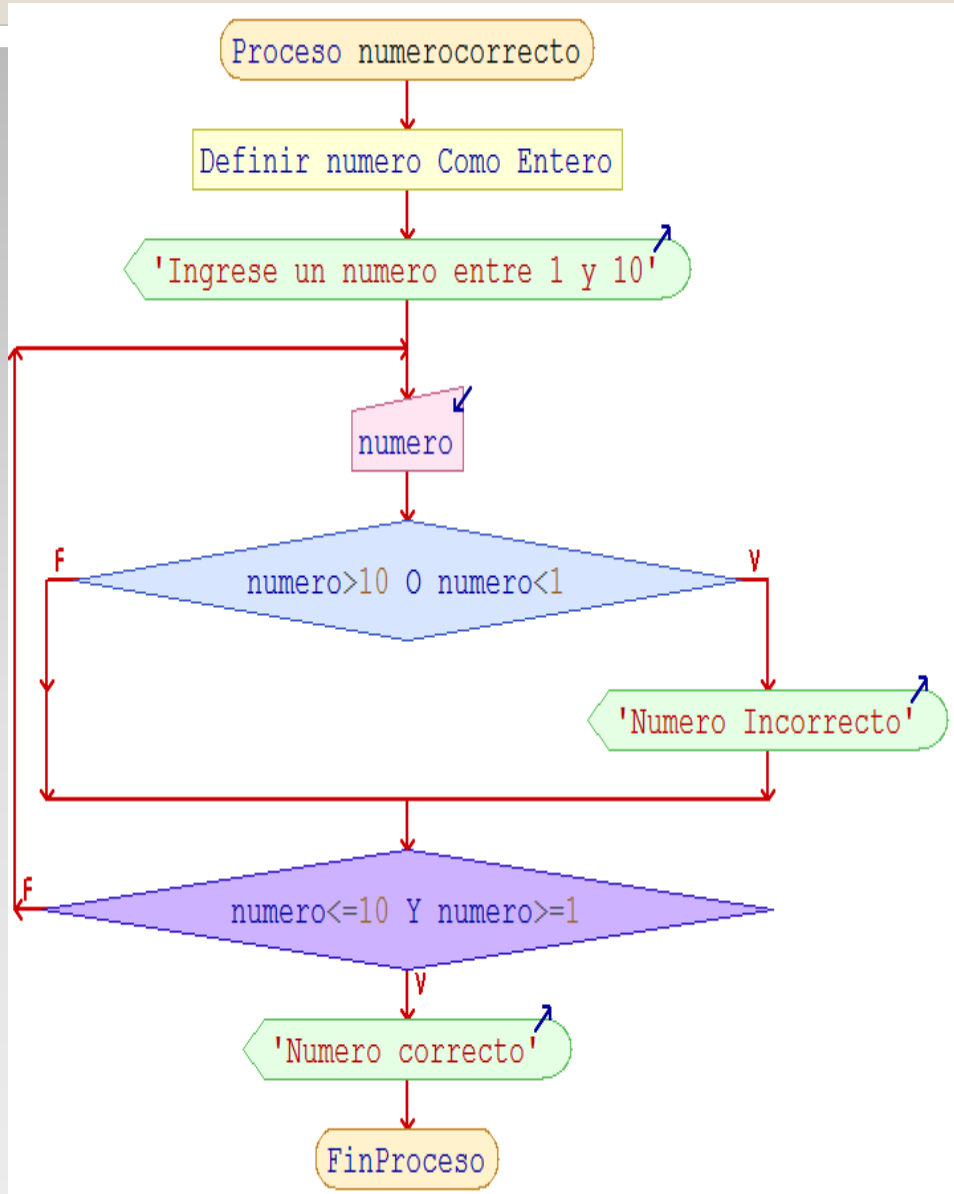
```
do
{
            enunciado1;
}while(expresión);
enunciado3;
```

Dentro del enunciado1 debe existir al menos una instrucción que haga que la condición sea falsa, de no ser así nunca saldrá de la proposición do..while.

1- se ejecutan los enunciados dentro del do
2- se evalúa la expresión si es cierta se vuelve al paso 1

```c
#include<stdio.h>

/* Ingresar un numero entre 1 y 10 */
int main()
{
    int numero;
    printf("Ingrese un numero entre 1 y 10\n");
    do
    {
        scanf("%i",&numero);
        if (numero>10 || numero<1)
        {
            printf("Numero Incorrecto\n");
        }
    }
    while (!(numero<=10 && numero>=1));
    printf("Numero correcto\n");
    return 0;
}
```



Proposición for

Esta proposición sirve para ejecutar de forma repetida durante un número finito de veces un conjunto de instrucciones. Se diferencia del while o do…while en que se conoce de antemano la cantidad de repeticiones.

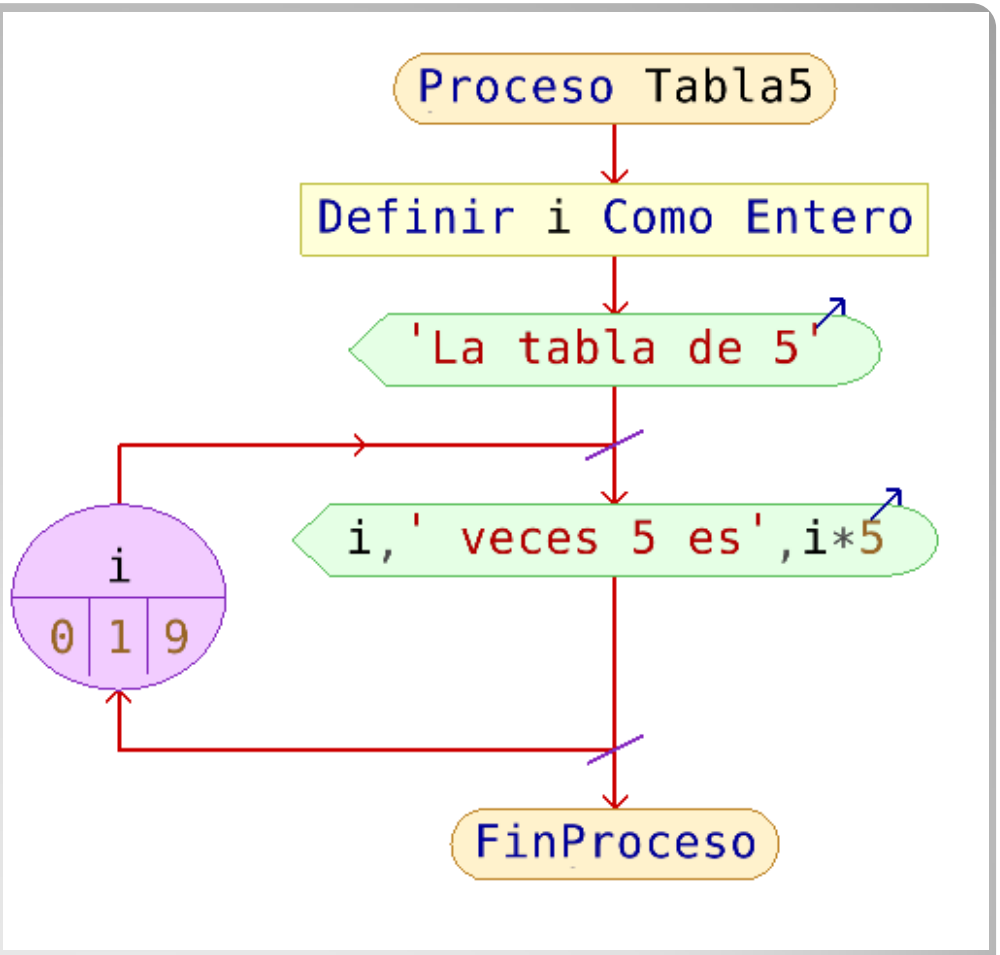
Su sintaxis es:

```
for(valor inicial;condición;incremento/decremento)
{
        enunciado1;
}
enunciado3;
```

1- se evalúa la condición inicial
2- se evalúa la condición
3- si es falsa el for termina y se ejecuta el enunciado3
4- si es cierta se ejecutan los enunciados dentro del for
5- se evalúa el incremento y se vuelve al paso 2

```c
#include<stdio.h>

int main()
{
int i;
printf("La table de 5\n");
for (i=0;i<=9;i++)
    {
     printf("%i veces 5 es %i\n",i,i*5);
     }
return 0;
}
```



Proposición switch

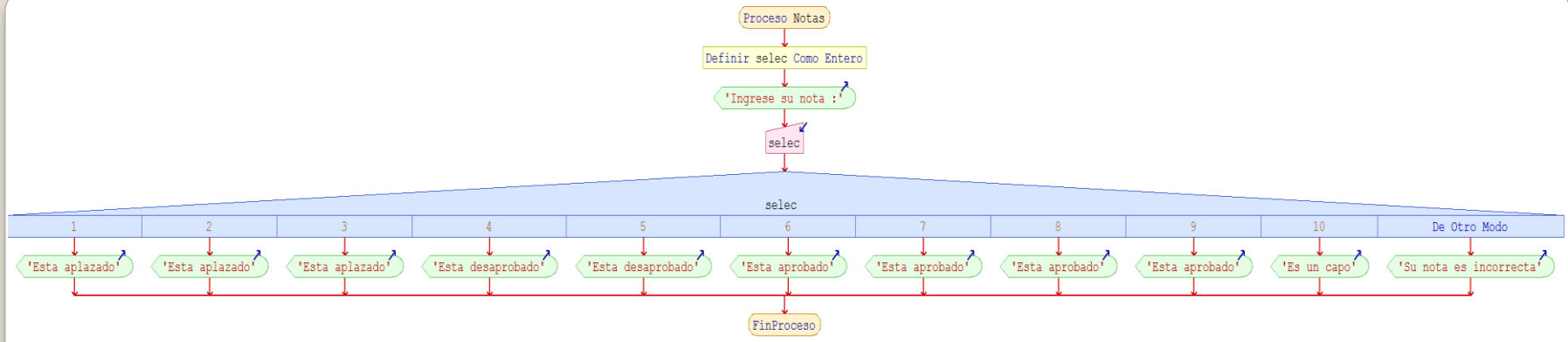
Permite que el programa ejecute distintos enunciados según el valor entero que puede tomar una expresión.

Su sintaxis es:

```
switch(expresión)
        {
        case plantilla 1:
        enunciado;
        break;
        case plantilla 2:
        enunciado;
        break;
        case plantilla 3:
        enunciado;
        break;
        .
        .
        default:
        enunciado;
        }
```

- La expresión puede ser cualquier valor entero (char, int, long, short).
- Es conveniente no omitir la opción de default.
- No olvidar los break en cada case.

1- se evalúa la expresión.
2- se lo compara con las plantilla ejecutándose el enunciado correspondiente
3- si no hay coincidencia se ejecuta el enunciado del default
4- se ejecuta el primer enunciado luego de la llave.




```
#include<stdio.h>
#include<conio.h>
int main()
{
int selec;
printf("Ingrese su nota :\n");
scanf("%i",&selec);
switch (selec)
{
case 1:
        printf("Esta aplazado\n");
        break;
case 2:
        printf("Esta aplazado\n");
        break;
case 3:
        printf("Esta aplazado\n");
        break;
case 4:
        printf("Esta desaprobado\n");
        break;
case 5:
        printf("Esta desaprobado\n");
        break;
case 6:
        printf("Esta aprobado\n");
        break;
case 7:
        printf("Esta aprobado\n");
        break;
case 8:
        printf("Esta aprobado\n");
        break;
case 9:
        printf("Esta aprobado\n");
        break;
case 10:
        printf("Es un capo\n");
        break;
default:
        printf("Su nota es incorrecta\n");
    }
getch)();return 0;
}
```

Héctor Albano Diego Codevilla

Enunciado break

Se puede colocar dentro de un for, while, do…while y switch. Al ejecutarse se da por terminado el enunciado. Si las sentencias estuvieran anidadas, solo actúa en la que la contiene.

Enunciado continue

Se lo puede colocar dentro de un for,while o do…while. Al ejecutarse comienza la siguiente iteración del ciclo que lo contiene. Los enunciados entre el continue y el final del ciclo no se ejecutan.

Proposición goto

El lenguaje C también permite la orden "**goto**", para hacer saltos incondicionales. **Su uso indisciplinado está muy mal visto**, porque puede ayudar a hacer programas llenos de saltos, difíciles de seguir. Se la considera innecesaria y contraproducente, ya que los programas son mas difíciles de mantener y seguir provocando muchos errores en la programación. Pero en casos concretos puede ser muy útil, por ejemplo, para salir de un bucle muy anidado (un "for" dentro de otro "for" que a su vez está dentro de otro "for": en este caso, "break" sólo saldría del "for" más interno). El formato de "goto" es:

goto donde;

y la posición de salto se indica con su nombre seguido de dos puntos (:)

donde:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int i, j;
    for (i=0; i<=5; i++)
        for (j=0; j<=20; j+=2)
        {
            if ((i==1) && (j>=7))
                goto salida;
            printf("i vale %d y j vale %d.\n", i, j);
        }
salida:
    printf("Fin del programa\n");
    return 0;
}
```

Función printf

La función **printf** (que deriva su nombre de "*print formatted*") imprime un mensaje por pantalla utilizando una "cadena de formato" que incluye las instrucciones para mezclar múltiples cadenas en la cadena final a mostrar por pantalla.

Sintaxis:

printf("texto %especificador de formato1 %especificador de formato2 \secuencia de escape",valor1,valor2) ;

Las comillas dobles siempre deben existir los datos adicionales son opcionales, pueden no aparecer.
Esta orden muestra un texto formateado en pantalla. Para usarla es necesario incluir <stdio.h> al principio de nuestro programa. Hemos visto cómo emplearla para mostrar número enteros, números reales y caracteres. Ahora vamos a ver más detalles sobre qué podemos mostrar y cómo hacerlo:

Especificadores de formato

c	Un único carácter
d	Un número entero decimal (en base 10) con signo
f	Un número real (coma flotante)
e	Un número real en notación exponencial, usando la “e” minúscula
E	Un número real en notación exponencial, usando la “E” mayúscula
g	Selecciona “e” o “f” (el más corto), con “e” minúscula
G	Selecciona “e” o “f” (el más corto), con “E” mayúscula
i	Un número entero decimal con signo
u	Un número entero decimal sin signo (unsigned)
h	Corto (modificador para un entero)
l	Largo (modificador para un entero)
x	Un número entero decimal sin signo en hexadecimal (base 16)
X	Un número entero decimal sin signo en hexadecimal en mayúsculas
o	Un número entero decimal sin signo en octal (base 8)
s	Una cadena de texto (que veremos en el próximo tema)

Secuencias de escape:

\n	Nueva línea (new line). El cursor pasa a la primera posición de la línea siguiente
\r	Retorno de carro (carriage return). El cursor pasa a la primera posición de la línea donde nos encontremos.
\t	Tabulador. El cursor pasa a la siguiente posición de tabulación.
\a	Alerta. Crea un aviso bien de forma visible o bien mediante sonido.
\b	Espacio atrás (backspace). Hace retroceder el cursor una posición a la izquierda.
\f	Alimentación de página (form feed). Crea una nueva página.
\\	Muestra la barra invertida.
\”	Muestra la comilla doble.
\?	Muestra un interrogante.
\número_octal	Muestra el carácter ASCII correspondiente según el número octal que se haya especificado.
\xnúmero_hexadecimal	Muestra el carácter ASCII correspondiente según el número hexadecimal que se haya especificado.
\v	Tabulación vertical.
\'	Apóstrofo o comilla simple.

Nota: %% para imprimir el signo de porcentaje

Además, los especificadores de formato pueden tener **modificadores**, que se sitúan entre el % y la letra identificativa del código.

- Si el modificador es un número, especifica la anchura mínima en la que se escribe ese argumento (por ejemplo: %5d).
- Si ese número empieza por 0, los espacios sobrantes (si los hay) de la anchura mínima se rellenan con 0 (por ejemplo: %07d).
- Si ese número tiene decimales, indica el número de dígitos totales y decimales si los que se va a escribir es un número (por ejemplo %5.2f), o la anchura mínima y máxima si se trata de una cadena de caracteres (como %10.10s).
- Si el número es negativo, la salida se justificará a la izquierda, dejando espacios en blanco al final (en caso contrario, si no se dice nada, se justifica a la derecha, dejando los espacios al principio).

Función scanf

Se utiliza para ingresar datos a través del teclado. Significa analizar con formato. La función lee datos del teclado de acuerdo con un formato específico y lo asigna a una o mas variables del programa.
Como ya sabemos, el uso de "scanf" recuerda mucho al de "printf", salvo porque hay que añadir el símbolo & antes de los nombres de las variables en las que queremos almacenar los datos. Aun así, los códigos de formato no son exactamente los mismos.
Como vemos, la diferencia más importante es que si en vez de un entero queremos un entero largo, se suele usar el mismo carácter escrito en mayúsculas.
Al igual que en "printf", se puede indicar un número antes del identificador, que nos serviría para indicar la cantidad máxima de caracteres a leer.
Por ejemplo, "scanf("%10s", &texto);" nos permitiría leer un texto de un tamaño máximo de 10 letras.

c	Un único carácter
d	Un número entero decimal (base 10) con signo
D	Un entero largo en base 10 sin signo
f	Un número real (coma flotante)
e,E	Un número real en notación exponencial
g,G	Permite “e” o “f”
i	Un número entero con signo
I	Un número entero largo con signo
u	Un número entero decimal sin signo (unsigned)
U	Un número entero decimal largo sin signo (unsigned)
h	Corto (modificador, para un entero)
l	Largo (modificador, para un entero)
x	Un número entero sin signo en hexadecimal (base 16)
X	Un número entero largo sin signo en hexadecimal
o	Un número entero sin signo en octal (base 8)
O	Un número entero largo sin signo en octal (base 8)
s	Una cadena de texto (que veremos en el próximo tema)