ESCUELA POLITÉCNICA SUPERIOR

INFORMÁTICA – CURSO 2024-25

PRÁCTICA 2. ESTRUCTURAS CONDICIONALES

HASTA AHORA...

En la práctica anterior se ha aprendido:

- La estructura principal de un programa en C: la función principal (`main`) y las librerías.
- Variables y Constantes: con sus respectivos tipos, usos y funcionamiento.
- Expresiones aritméticas utilizadas para operar con las variables.

OBJETIVOS

En esta práctica aprenderemos el uso de las funciones de entrada y salida de datos (`printf` y `scanf`, respectivamente) y la estructura condicional (`if/else`), con sus respectivas expresiones lógicas para poder evaluar las diversas condiciones.

PRERREQUISITOS

Para el correcto desarrollo de esta práctica, el alumno ANTES DE LA SESIÓN DE PRÁCTICA EN EL LABORATORIO, debe leer con atención y comprender el contenido de la parte teórica de dicha práctica. Además, el alumno habrá trabajado los conceptos relacionados con la práctica anterior, además de haber completado los ejercicios correspondientes en casa.

Se aconseja el uso del material de apoyo de la asignatura. Así, para esta práctica el alumno puede consultar la siguiente bibliografía:

- Rodríguez Jódar, M.A. y otros, "Fundamentos de Informática para Ingeniería Industrial":
    - *Capítulo 4, Fundamentos de Programación*: apartado 4.3, *Estructuras Condicionales.*

***Primera parte:* funciones printf y scanf.**

¿Qué es una función? Aunque se profundizará en este aspecto en la práctica 4 (dedicada íntegramente a funciones), es necesario comentar su uso y definición de cara a poder entender los conceptos que se detallan en esta primera parte.

Así pues, una función es un **conjunto de instrucciones que llevan a cabo una tarea específica y que se empaquetan dentro de un contenedor (o caja negra) para que el usuario no tenga necesidad de conocer cómo está implementada** internamente; únicamente debe saber cómo se la invoca para que se ejecute. De esta forma, conociendo su nombre y los argumentos (datos) que hay que aportarle para que se ejecute correctamente, podemos "llamar" a una función ya implementada por otro usuario. Además, **una de las características más importantes de las funciones es la reutilización**, ya que se las puede llamar en el programa tantas veces como se necesiten… **¡sin necesidad de copiar y pegar el código interno de cada una!** (que es lo que se hubiera hecho si no fueran funciones independientes)

Este es el caso de las funciones *printf* y *scanf*: no es necesario conocer cómo se han hecho (se podría ver dentro de la librería *stdio.h*, por ello hay que incluirla en nuestro programa si se hace uso de estas funciones); simplemente con saber su nombre y los datos a aportar se podrían ejecutar sin problema. Así pues, se pasará a describir la nomenclatura utilizada para ejecutar dichas funciones.

En la práctica anterior hicimos un uso limitado de la función `printf` para mostrar datos por pantalla. La definición formal de la función `printf` es la siguiente:

```
printf("texto_de_formato", lista_argumentos);
```

Entre las comillas (“) podemos escribir la frase que queramos que la función *printf* muestre en pantalla. Pero ¿se puede mostrar únicamente texto estático o se pueden “incrustar” valores de variables? **SÍ a ambas cosas**.

- Mostrar texto estático: no se utiliza la “lista_argumentos”, únicamente se hace uso del “texto_de_formato”. Se añade entre comillas el texto a mostrar. Ejemplo:

```
printf(“Este es un mensaje de prueba”);
```

- Mostrar mensaje con valores de variables: hay que tener en cuenta dos aspectos.

  - En la “lista_argumentos” se indican las variables cuyos valores se van a mostrar, separadas por comas y por **estricto orden de aparición**.

  - En el “texto_de_formato” se incrustan identificadores para mostrar la posición donde se situarán los valores de las variables. El identificador estará relacionado con el tipo de variable utilizada. Estos identificadores están precedidos por el carácter ‘%’ y son los siguientes:

CÓDIGO	FORMATO
%d	**Muestra un valor entero con signo (int, char, etc.)**
%u	*Muestra un valor entero sin signo (unsigned int, etc.) **
%x	*Se escribe en pantalla un número hexadecimal **
%f	**Se muestra un número real de tipo float**
%lf	*Se muestra un número real de tipo doublé **
%c	Muestra un carácter de la tabla ASCII **
%s	Cadena de caracteres **

***** ***Prácticamente no se van a utilizar, salvo casos puntuales.***
****** ***Se utilizarán en posteriores prácticas.***

Ejemplos:

```
int v;
float v2;
printf(“La variable tiene el valor %d”, v);
printf(“Las variables tienen los valores %d y %f”, v, v2);
```

Ejemplo completo:

```
#include <stdio.h>      //Importante!! Esta librería es la que contiene
                        // las funciones printf y scanf
main()
{
    int a;
    float b;

    a = -33;

    printf(“Esto muestra un entero: %d\n”, a);
    printf(“Esto muestra una suma entera: %d\n”, a+120);
    printf(“Esto muestra el mismo valor como entero y natural: %d, %u\n”, a, a);
    printf(“Ahora mostramos un valor real: %f\n”, b);
}
```

Se puede observar que al final de cada frase hemos escrito el carácter ‘\n’. Éste provoca que, después del texto indicado en el *printf*, se realice un salto de línea. Este tipo de caracteres o códigos se denominan s*ecuencias de escape*. Las que se

utilizan se resumen en la siguiente tabla:

CÓDIGO	SIGNIFICADO
\a	Carácter de alarma
\b	Retroceso
\n	Salto de línea
\t	Tabulador horizontal
\r	Retorno de carro
\"	Dobles comillas
\'	Comillas simples
\\	Barra invertida
\?	Signo de interrogación

Por otro lado, para el proceso inverso se puede utilizar la función "scanf". Esta función permite recoger un valor introducido desde teclado y modificar la variable indicada con el valor que ha recogido. La sintaxis es la siguiente:

```
scanf("texto_de_formato", &variable);
```

RECOMENDACIONES:

- En el caso de *scanf* se recomienda indicar en el "texto_de_formato" **únicamente** los identificadores de las variables cuyos valores se quieren recoger. Estos identificadores son idénticos a los utilizados en el *printf* y, al igual que antes, dependen del tipo de variable.
- Si se pretenden recoger valores de diversas variables, se recomienda utilizar una llamada a *scanf* de manera individual para cada una.

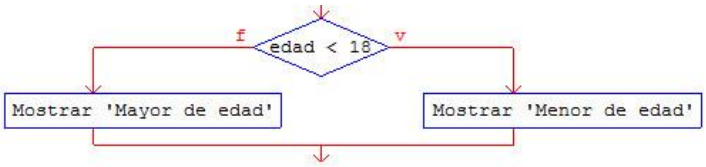
Ejemplo completo:

```
#include <stdio.h>
main() {
      float b;
      scanf("%f", &b);
      printf("El valor tecleado ha sido: %f\n", b);
}
```

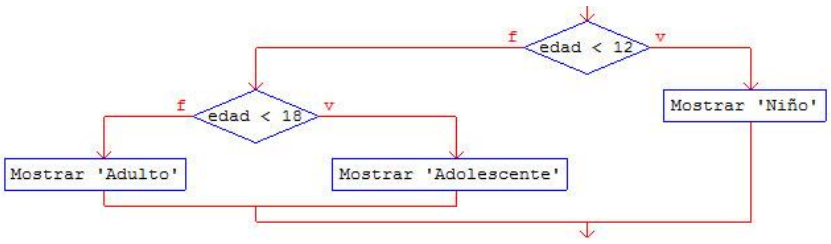
Impotante: El uso del símbolo '&' (Alt Gr. + 6) delante del nombre de la variable es obligatorio en el *scanf*. Si olvidamos ponerlo puede ocurrir un error grave durante la ejecución de la aplicación.

***Segunda Parte:* Estructuras condicionales (if / else).**

Hasta ahora sólo hemos podido escribir programas que se ejecutan de manera secuencial (línea a línea), incapaces de tomar decisiones o de variar su ejecución atendiendo a los diversos eventos que se vayan produciendo en el programa. Para poder realizar unas acciones u otras en función de ciertas condiciones usamos la estructura condicional.
El siguiente diagrama de flujo muestra un ejemplo de una estructura condicional. El rombo identifica la estructura condicional y la expresión que encierra establece la condición. En función del resultado de evaluar dicha condición, la ejecución tomará la rama de la condición verdadera (identificada por 'v' en el diagrama) si se cumple la condición o la rama falsa (identificada por 'f') en el caso contrario. En este ejemplo, si 'edad' es menor de 18 tomará la rama verdadera por lo que mostrará por pantalla 'Menor de edad'.



Las estructuras condicionales pueden anidarse para dar lugar a múltiples ramas condiciones. Si suponemos en el siguiente ejemplo que la edad es de 15 años, la primera condición tomará la rama falsa evaluando así la segunda condición. La segunda condición tomará la rama verdadera por lo que se mostrará por pantalla 'Adolescente'.



En lenguaje C las estructuras selectivas se realizan fundamentalmente con las instrucciones if/else y su sintaxis es la siguiente:

```
if (condición 1)
{     /* Si se cumple la condición 1*/
      Bloque de instrucciones.
}
else if (condición 2)
{     /* Si no se ha cumplido la condición 1 y sí la condición 2*/
      Bloque de instrucciones.
}
else if (condición 3)
{     /* Si no se ha cumplido ni la 1, ni la 2 y se cumple la 3 */
      Bloque de instrucciones.
}
…
else if (condición n)
{     /* Si no se han cumplido las anteriores y se cumple la n */
      Bloque de instrucciones.
}
else
{     /* Si NO se cumple ninguna de las anteriores. */
      Bloque de instrucciones.
}
```

La ejecución de la estructura anterior consistiría en evaluar la *condición 1*, si se cumple dicha condición, ejecuta el bloque de instrucciones ubicado dentro de esa primera rama y finaliza la ejecución de la estructura. Si no se cumple, salta a evaluar la *condición 2*, ante la que se procedería de igual forma. Por último, si no se cumple ninguna de las condiciones anteriores, el programa toma la rama del último "else" (obsérvese que no tiene condición específica), en el que convergen todos los casos que no satisfacen ninguna de las condiciones anteriores.

Nótese que la ejecución de las ramas es exclusiva, esto es, que únicamente se va a ejecutar una de estas ramas. Si se satisfacen más de una condición, se tomaría la rama situada antes por orden estricto.

Además, hay que tener en cuenta que la parte de la cláusula *else* es opcional, así que podríamos escribir algo como:

```
if (condición)
{
      Bloque de instrucciones
}
```

La diferencia estaría en que en el caso de que no se cumpla la condición, la sentencia *if* no haría nada; únicamente se seguiría la ejecución del programa con las instrucciones que hubieran a continuación de la estructura condicional.

El diagrama de flujo anterior se corresponde con la estructura condicional del siguiente programa:

```
#include <stdio.h>
main() {
      int edad;
      printf("Introduzca su edad: ");
      scanf("%d", &edad);
      if (edad < 12)
            printf("\n Niño");
      else  if (edad < 18)
            printf("\n Adolescente");
```

```
        else
              printf("\n Adulto");
}
```

Pero… ¿cómo se indican las condiciones en C?

La forma más simple de condición son las llamadas *expresiones relacionales*. Éstas comparan dos operandos y devuelven un resultado que puede ser *verdadero* o *falso*. Esos operandos son expresiones aritméticas (desde una variable o valor, hasta una expresión compleja).

En el siguiente cuadro resumimos las expresiones relacionales usadas en lenguaje C:

Expresión RELACIONAL	**Resultado de la expresión**
Operando1 **>** *Operando2*	*Es verdadero* si *Operando1* **ES MAYOR QUE** *Operando2*
Operando1 **>=** *Operando2*	*Es verdadero* si *Operando1* **ES MAYOR O IGUAL QUE** *Operando2*
Operando1 **<** *Operando2*	*Es verdadero* si *Operando1* **ES MENOR QUE** *Operando2*
Operando1 **<=** *Operando2*	*Es verdadero* si *Operando1* **ES MENOR O IGUAL QUE** *Operando2*
Operando1 **==** *Operando2*	*Es verdadero* si *Operando1* **ES IGUAL QUE** *Operando2*
Operando1 **!=** *Operando2*	*Es verdadero* si *Operando1* **ES DISTINTO QUE** *Operando2*

Ejemplo:

```
if (nota>5) {
      printf("Has aprobado con un %f\n", nota);
}
```

Podemos combinar varias expresiones relacionales mediante los operadores lógicos existentes en álgebra de Boole (rama del álgebra que estudia las operaciones relacionales), construyendo *expresiones lógica*s más complejas:

Expresión LÓGICA	**Resultado de la expresión**
Operando1 **&&** *Operando2*	*Es verdadero* si *Operando1* es verdadero **Y** *Operando2* es verdadero
Operando1 **||** *Operando2*	*Es verdadero* si *Operando1* es verdadero **O** *Operando2* es verdadero
! *Operando1*	*Es verdadero* si *Operando1* **NO** es verdadero

Ejemplo:

```
if (nota>=7 && nota<9) {
      printf("Notable");
}
```

Los operandos pueden ser expresiones relacionales más complejas. Para finalizar, al igual que las operaciones aritméticas, las expresiones lógicas y relacionales tienen su precedencia.

Mayor ... Menor

Expresiones aritméticas	<, <=, > y >=	== y !=	&&	||

Ejercicio 1: Analice el siguiente código:

```
int main(void) {
	int a, b, c=9, d;
	printf(“Introduzca el valor de a (entero): ”);
	scanf(“%d”, &a);
	printf(“Introduzca el valor de b (entero): ”);
	scanf(“%d”, &b);
	d = a + b + c;
	printf(“El resultado de sumar a + b + c es igual a %d.\n”, d);
	return 0;
}
```

¿Qué hace el programa? ¿Cuál sería el resultado de “d” suponiendo que el usuario introduce por teclado los valores 4 y 6 para a y b respectivamente?

Ejercicio 2: Rellene los huecos del siguiente código para que lea por teclado las 2 notas de un alumno y muestre por pantalla la nota final obtenida como la media aritmética de ambas notas.

```
int main(void){
	float n1,n2, final;
	scanf("______", &n1);
	scanf("%f", _______);
	final = ( _____ + ______) _______;
	printf("La nota final es ______ ", _________);
	return 0;
}
```

Ejercicio 3: Escriba un programa que pida por teclado un valor de tipo *float*, que será el radio de una circunferencia. A continuación, debe mostrar la longitud de la circunferencia y su área.

Ejercicio 4: Realice un programa que muestre por pantalla el valor de la pendiente de una recta dada por 2 puntos P1=(x1,y1) y P2=(x2,y2), siendo los valores de las coordenadas introducidos por teclado (utilice el tipo de dato que considere oportuno). Recuerde que la pendiente de una recta dada por los puntos P1y P2 puede calcularse como:

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

Ejercicio 5: Complete el siguiente programa encargado de leer desde teclado la edad de una persona e indicar si dicha persona es mayor de edad o no.

```
int main(void) {
	int edad;
	printf(“Introduzca tu edad: ”);
	scanf(“%d”, &edad);
	if ( ____________________________________ ) {
		printf (“Con _______ años eres menor de edad”, _______)
	}
	else {
		printf (“Con _______ años eres mayor de edad”, _______)
	}
	return 0;
}
```

Ejercicio 6: Analice el siguiente código:

```
int main(void) {
	int li, ld, v;
	printf(“Introduzca el valor de li (entero): ”);
	scanf(“%d”, &li);
	printf(“Introduzca el valor de ld (entero): ”);
	scanf(“%d”, &ld);
	printf(“Introduca el valor de v (entero): “);
	scanf(“%d”, &v);
	if(v >= li && v < ld)  {
		printf(“Dentro \n”);
	}
	else if(v < li)  {
		printf(“Menor \n”);
	}
	else {
		printf(“Mayor \n”);
	}
	return 0;
}
```

Explique qué hace el programa y proponga conjuntos de valores de li, ld, y v para cubrir cada caso:

Caso número:	Valor “li”	Valor “ld”	Valor “v”	Resultado

Ejercicio 7: Rellene los espacios en blanco del siguiente código para que muestre por pantalla si la temperatura especificada por teclado corresponde a un ambiente frío, agradable o caluroso suponiendo que a partir de 21 grados (inclusive) es agradable y a partir de 32 (no inclusive) es caluroso.

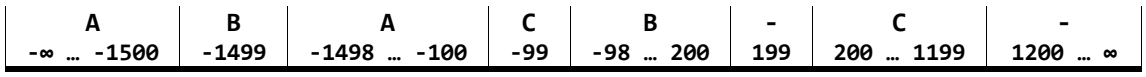
```
int main(void)
{	int temp;
	scanf("____", ___temp);
	printf("La temperatura _____ se corresponde a un ambiente ", ______);
	if (                )  {
		printf("frío")
	}
	else if (                )  {
		printf("agradable")
	}
	else {
		printf("caluroso")
	}
	return 0;
}
```

Ejercicio 8: Sea el siguiente código:

```
int main(void) {
	int v;
	scanf("%d", &v);
	if ( v < -2000) {
		printf("A");
	}
	else if ( (v>0 && v<100) || (v>200 && v<300)) {
		printf("B");
	}
	else if ( (v>0 && v<300)) {
		printf("C");
	}
	else if (  v<0 || ( v>=500 && v<1000)) {
		printf("D");
	}
	else if ( v == 0) {
		printf("E");
	}
	else {
		printf("F");
	}
	return 0;
}
```

Indique sobre la siguiente recta real, el rango de valores de la variable v para los que se ejecutaría cada rama de la estructura condicional indicada. Sirva de ejemplo el rango de valores de la rama A.

Ejercicio 9: Complete el siguiente programa para que muestre por pantalla A, B, C o no mostrar nada (‘-‘) en función de un valor entero introducido por teclado según la siguiente recta real:




```
int main(void) {
	int v;
	scanf("%d", &v);
	if ( v <= -100 && v!=-1499) {
		printf("A");
	}
	else if ( ____________________________ ) {
		printf("B");
	}
	else if ( ____________________________ ) {
		printf("C");
	}
	return 0;
}
```

Ejercicio 10: Escriba un programa que calcule la nota final de un alumno a partir de su nota de teoría y de prácticas (ambas como valores reales) suponiendo que la nota final se obtiene del 40% de la nota de teoría y del 60% de la nota de práctica. El programa deberá solicitar ambas notas por teclado incluyendo los mensajes que indiquen al usuario lo que se le solicita (por ejemplo, “Introduzca la nota de teoría:”) y mostrar la nota final calculada.

Ejercicio 11: Amplíe el programa anterior para que verifique si las notas introducidas son válidas (están comprendidas entre 0 y 10). En caso de que alguna de ellas no fuese correcta, debe mostrar el mensaje “Error: alguna de las notas es incorrecta”.

Ejercicio 12: Complete el programa anterior para que muestre por pantalla el mensaje "SUSPENSO", "APROBADO", "NOTABLE" o "SOBRESALIENTE" en función de la nota final obtenida.

Ejercicio 13: Atendiendo a la siguiente clasificación de sustancias según su nivel de *pH* y suponiendo que nos focalizamos únicamente en dicho subconjunto, se pide lo siguiente:

Algunos valores comunes del pH

Sustancia/Disolución	pH
Disolución de HCl 1 M	0,0
Jugo gástrico	1,5
Refresco de cola	2,5
Cerveza	4,5
Té	5,5
Leche	6,5
Agua pura	7,0
Agua de mar	8,0
Amoníaco	11,5
Hidróxido sódico	14

a) Realice un programa que pida por teclado un valor real que represente el nivel de *pH*. El programa deberá indicar mediante un mensaje de texto a cuál de los compuestos contemplados pertenece.

b) Suponga que se pueden aportar valores que no se correspondan con ninguna de las sustancias anteriores. Así pues, realice un programa que contemple esta problemática y, en dicho caso, debe indicar entre qué dos sustancias se encuentran.

Ejercicio 14: detecte y corrija los posibles errores que pueda contener el siguiente código:

```
int main(void){
	int pH;
	float pH;
	double tolerancia;
	printf("Introduzca el valor de pH: ");
	scanf("%f", &li);
	pH = pH + tolerancia;
	pH = pH – tolerancia;
	printf("Rango de valores aceptados de pH: [%d  %d]", pH, pH);
	return 0;
}
```

Por ejemplo, si el valor de pH introducido es 3.1, y el valor de rango es 0.4, el mensaje mostrado por pantalla sería: "Rango de valores aceptados de pH: [3.5 2,7]"

Ejercicio 15: Escriba un programa que resuelva una ecuación de segundo grado ($ax^2 + bx + c = 0$). Para ello:

a) Introduzca por teclado los datos necesarios (a, b y c) y muestre las dos soluciones. Recuerde que las soluciones se obtienen de la siguiente fórmula:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Utilice la función ***sqrt****(x)* para calcular la raíz cuadrada de x. Para utilizar dicha función, incluya la siguiente línea arriba del código: #include <math.h>

b) Como ha comprobado, no tenemos en cuenta casos en los que la ecuación de segundo grado no tiene soluciones reales (contenido de la raíz cuadrada con valor negativo). Téngalo en cuenta ahora, comprobándolo antes de mostrar los resultados. En caso de no haber solución, mostrar un mensaje que lo indique; en caso opuesto, muestre las soluciones al igual que en el apartado anterior.

Ejercicio 16: Realice un programa que lea del teclado una temperatura en grados Centígrados, calcule los grados Fahrenheit y escriba por pantalla el deporte que es apropiado practicar a esa temperatura, teniendo en cuenta la siguiente tabla:

DEPORTE	TEMPERATURA en grados Fahrenheit
Natación	> 85
Tenis	70 < TEMP <= 85
Golf	35 < TEMP <= 70
Esquí	32 < TEMP <= 35
Marcha	<= 32

Para convertir grados Centígrados (variable c) a Fahrenheit (variable f) utilice ecuación:

$$f = c \cdot \frac{9}{5} + 32$$

Ejercicio 17: Escribir un programa que, dada una fecha introducida por teclado (día y mes en formato numérico), muestre por pantalla la estación del año a la que pertenece. Las fechas de inicio de las estaciones son:

Primavera: *21 de marzo*
Verano: *21 de junio*
Otoño: *23 de septiembre*
Invierno: *22 de diciembre*

Por ejemplo, la fecha corresponderá a la primavera si es marzo y día mayor o igual a 21, si es abril, si es mayo o si es junio y día menor a 21.

Ejercicio 18: Analice y complete el siguiente programa encargado de realizar operaciones aritméticas. El programa solicitará el primer operando, posteriormente el signo de la operación a realizar (+,- ,* o /) y finalmente el segundo operando. El programa mostrará el resultado de la operación correspondiente o "Error" en caso de introducir un signo distinto.

```
int main(void)
{     float op1, op2, res;
      char signo;
      scanf(__________, &op1);
      scanf(%c, &signo);
      scanf("%f", _____);
      if (________ == '+') {
            res = op1 + op2;
            printf("= ____________________________________________);
      }
      else if ( signo == ______) {
            res = _________________;
            printf("= ____________________________________________);
      }
      else if ( _________________) {
            res = _________________;
            printf("= ____________________________________________);
      }
      else if ( _________________) {
            res = _________________;
            printf("= ____________________________________________);
      }
      else {
            printf(_____________________________);
      }
      return 0;
}
```

ANEXO: ESTRUCTURA CONDICIONAL SWITCH

OBJETIVOS

El objetivo de este anexo es aprender el uso de la estructura condicional *switch*, que puede utilizarse en lugar del clásico *if*/*else*.

La estructura switch
Está formada por una primera sentencia donde se indica la expresión a evaluar como condición:

```
switch (expresión)
{
      … //cuerpo del switch
}
```

En segunda instancia, dentro del cuerpo del switch, se evalúan todos los posibles valores que puede admitir la expresión. Cada una de estas posibilidades tiene la siguiente nomenclatura:

```
case valor:
      //Bloque de instrucciones
      break;
```

Por último, tras implementar todos los posibles casos dentro del cuerpo del switch, es obligatorio indicar un caso final (similar a *else* con el que terminaban muchas sentencias condicionales vistas anteriormente), que posee la siguiente nomenclatura:

```
default:
      //Bloque de instrucciones
      break;
```

En este último caso, el bloque de instrucciones puede estar vacío (al igual que el *else* podía contemplarse o no en las estructuras condicionales anteriores). Sin embargo, en la estructura *switch* el uso de la sentencia *default* es obligatorio.

¿Por qué utilizar la instrucción "break"? ¿Eso qué hace? Muy simple: la ejecución de las diversas ramas de la estructura *if*/*else* eran exclusivas; esto quería decir que únicamente se ejecutaba una de ellas. El funcionamiento del *switch* no es exactamente así, sino que se accedería al caso concreto que se corresponda con el valor buscado y se ejecutaría; sin embargo, tras él se ejecutarían todos los casos situados a continuación de este. Para evitar esto, se hace uso de la instrucción *break*, que fuerza a salir del bloque *switch*.
Un ejemplo concreto:

```
switch (valorIntroducido){
      case 0: //Evalúa la condición valorIntroducido == 0
            printf("Has metido el valor 0\n");
            break;
      case 1: //Evalúa la condición valorIntroducido == 1
            printf("Has metido el valor 1\n");
            break;
      case 2: //Evalúa la condición valorIntroducido == 2
            printf("Has metido el valor 2\n");
            break;
      default: //No se cumple ninguna de las anteriores
            printf("Has metido algo muy raro\n");
            break;

}
```