Tema 2. Condicionales.

1. Condiciones.

   Una condición es una expresión booleana, es decir, que devuelve *true* o *false*.

2. Estructuras if e if-else.

   La estructura if tiene la forma

   ```
   if (condición) {
     acciones;
   }
   ```

   y se traduce en que las *acciones* se ejecutarán si la *condición* se evalúa como *true*. En caso contrario se saltarán y se seguirá ejecutando el programa a partir de la siguiente línea al bloque *if*.

   La sentencia if – else tiene la forma

   ```
   if (condición) {
     acciones;
   } else {
     otras_acciones;
   }
   ```

   Su efecto es el siguiente: si la *condición* se evalúa como *true*, se ejecutarán las *acciones* y a continuación se seguirá ejecutando el programa a partir del final del bloque *if – else*. Si la *condición* se evalúa como *false*, se ejecutarán las *otras_acciones* y el programa continuará después del bloque *if – else*.

3. Estructura if- else if.

   Las sentencias if e if -else se pueden anidar:

   ```
   if (condición1) {
     if (condición2) {
       acciones1;
     } else {
       acciones2;
     }
   }
   ```

   Además, si la acción consta de una única sentencia, se pueden omitir las llaves (en otras palabras, no es necesario definir un bloque). Aprovechando esta característica, cuando tenemos una serie de condiciones excluyentes, podemos anidar sentencias *if – else* de la siguiente forma:

```
if (condición1) {
  acciones1;
} else if (condición2) {
  acciones2;
} else if (condición3) {
  acciones3;
.
.
}
```

Escribiendo todas las sentencias *if* en el mismo nivel de identación, evitamos que las líneas se nos vayan a la derecha, mejorando la legibilidad del código, aunque realmente estén anidadas.

Es muy importante que las condiciones sean excluyentes, es decir, que no haya ningún caso (ninguna combinación de valores) que permita evaluar como *true* más de una condición.

4. Estructura switch.

Cuando tenemos una decisión múltiple basada en los posibles valores de una expresión, en lugar de usar la estructura anterior podemos usar la estructura *switch* de la siguiente forma:

```
switch (expresión) {
  case valor1:
    acciones1;
    break;
  case valor2:
    acciones2;
    break;
  .
  .
  default:
    acciones_por_defecto;
}
```

Su efecto es el siguiente: se evalúa la *expresión*, que debe ser de tipo entero, *String* o enumerado (los tipos enumerados se verán más adelante) y se busca una clausula *case* cuyo valor coincida con el resultado de la *expresión*, en orden de aparición. Si se encuentra una coincidencia, se ejecutan las acciones correspondientes a dicha clausula *case*. Si no se encuentra ninguna coincidencia y se ha especificado una clausula *default*, se ejecutarán las acciones por defecto.

La clausula *default* es opcional, y en caso de aparecer tiene que ser la última.

Las sentencias *break* sirven para que una vez ejecutadas las acciones de una clausula case, salte la ejecución a la línea siguiente al final del bloque *switch*.
En algunos casos, nos encontramos con casos equivalentes, es decir, varios valores que deben producir el mismo resultados. En esas circunstancias podemos encadenar varias clausulas case sin separarlas por *break*, para que todas ellas ejecuten las mismas acciones.

5. Expresiones condicionales.

Existe un tipo de expresiones llamado expresión condicional o expresión ternaria, ya que consta de tres operandos, y tiene la sintaxis

```
condición?expresión1:expresión2
```

Cuando aparece una expresión de este tipo, se evalúa la *condición*, y si el resultado es *true*, se devuelve el resultado de evaluar la *expresión1*. En caso contrario se devuelve el resultado de evaluar *expresión2*.