Sentencias de toma de decisiones

Ha llegado la hora de introducir uno de los puntos más importantes de toda la programación. La toma de decisiones, cuando realizar una acción u otra, y como realizar esta acción, si tiene que ser en bucle o una condición nos indicara si realizarla.

Índice

1 - Sentencias Condicionales

Son las que realizamos cuando tenemos que tomar una decisión basándonos en cierta lógica, una forma de verlo gráficamente sería. Imaginémonos que vamos por una carretera y esa carretera tiene un desvío, si tomamos el desvío sería como entrar en una sentencia condicional.

1.1 - If -else

La primera sentencia condicional que vamos a ver es la sentencia `If-Else` la cual literalmente significa "sí-sino" por lo tanto la sentencia condicional va enlazada con un **operador lógico** y se representa de la siguiente forma

```
if (A < B)
{
    Console.WriteLine("A es menor que B");
}
else
{
    Console.WriteLine("A no es menor que B");
}
```

Si el contenido dentro de la sentencia if es una sola línea, no necesitamos utilizar los paréntesis, pero personalmente lo recomiendo. Ya que así es más fácil leerlo después.

```
if (A < B)
    Console.WriteLine("A es menor que B");
```

```
else
    Console.WriteLine("A no es menor que B");
```

Des ...on
llan ...ue
der ...ar
a hacer que te pierdas si estas leyendo el código. En el trabajo he tenido varios casos en los que tardas mas tiempo en saber que hace tanto if dentro de tanto if que en arreglar el propio problema en sí.

```
A < B ? Console.WriteLine("A es menor que B") :  Console.WriteLine("A no es menor
```

En caso de que solo quieras incluir la opción de dentro del `If` porque el else va a seguir el curso normal de la ejecución no necesitas añadir el `else` .

```
if (A < B)
{
    Console.WriteLine("A es menor que B");
}
```

1.2 - Swich-case

Muchas veces nos encontramos con que escribimos muchas sentencias IF segudias una detras de otra. para solucionar este problema tenemos la implementación de las sentencias swich.

una sentencia swich coge una variable y la analiza por cada uno de los casos disponibles, como vemos en el ejemplo indicamos `swich(variable)` para analizar la variable. y esta puede ser de cualquier tipo, no tiene porque ser un string, puede ser un numero, una fecha, etc.

```
string mes = "enero";
switch (mes)
{
    case "enero":
        Console.WriteLine("enero tiene 31 dias");
        break;
    case "febrero":
        Console.WriteLine("febrero tiene 28 dias");
        break;
    default:
        Console.WriteLine("mes no encontrado");
        break;
 }
```

Como vemos en el ejemplo, analizamos cada case y al final del mismo introducimos la sentencia break; la cual nos hara salir del bucle. Para este ejemplo hemos utilizado únicamente dos meses, con lo que la sentencia es bastante pequeña pero en caso de poner todos los meses, sería mucho mayor.

Como todo el mundo sabe, los meses (a excepción de febrero) tienen 30 o 31 dias, la sentencia swich nos permite agrupar en caso de que queramos la misma salida para varios de ellos

```
string mes = "enero";
switch (mes)
{
    case "enero":
    case "marzo":
    case "mayo":
    case "julio":
    case "agosto":
    case "octubre":
    case "diciembre":
        Console.WriteLine("enero tiene 31 dias");
        break;
    case "febrero":
        Console.WriteLine("febrero tiene 28 dias");
        break;
    case "abril":
    case "junio":
    case "septiembre":
    case "noviembre":
        Console.WriteLine("febrero tiene 28 dias");
    default:
        Console.WriteLine("mes no encontrado");
        break;
 }
```

Con C# 8 su funcionalidad ha aumentado, puedes darles una ojeada **aquí.**

2 - Sentencias Iteracionales - Bucles

Comúnmente cuando estamos programando nos encontramos con que tenemos una lista de objetos y a todos ellos (o a una mayoría aplicando sentencias condicionales) debemos calcular algún valor o imprimir al usuario distinta información. para ello en vez de hacer cada caso uno por uno, utilizamos bucles, que nos permiten iterar sobre cada uno de los elementos de una lista.

2.1 - Bucle For

El bucle for es el bucle mas implementado, y se utiliza para reptir acciones cierto numero de veces

```
for (int i = 0; i < 10 ; i++){
    Console.WriteLine("Iteración número "+i);
}
```

Como vemos cuando declaramos el bucle for, lo hacemos junto con:

- Declaración e inicialización de la variable: `int i = 0;`
- Expresión lógica para comparar si se tiene que ejecutar: `i < 10;`
- Operador aritmético de incremento en una unidad: `i++;`

2.2 - Bucle while

Evaluamos la expresión lógica entre paréntesis y ejecutamos el bloque si es verdadera, al terminar volvemos a evaluar la expresión, y si es válida volvemos a ejecutar el bloque.

```
int contador = 0;
while(contador < 10){
      Console.WriteLine("Iteración número "+i);
      contador++;
}
```

2.3 - Bucle do-while

Similar a la anterior, solo que en este caso el bloque se ejecuta al menos una vez y posteriormente sigue ejecutando mientras la expresión sea cierta.

```
int contador = 0;
do{
    Console.WriteLine("Iteración número "+contador);
    contador++;
}while(contador < 10);
```

Hay que tener mucho cuidado cuando creamos bucles ya que es posible -sobre todo cuando estamos aprendiendo - hacer bucles infinitos. los cuales, si suceden en nuestra aplicación cuando estamos desarrollando, nos dará un "out of memory" porque el ordenador se queda sin memoria y ya está, no pasa nada.

pero si lo realizamos en producción, causaría que la aplicación dejará de funcionar y será un problema mucho mayor.

una forma de crear un bucle infinito es la siguiente:

```
int contador = 11;
do{
```

```
    Console.WriteLine("Iteración número "+contador);
    contador++;
}while(contador > 10);
```

Cor ue el bucle no dejará de ejecutarse nunca. Obtendremos el mismo resultado si por ejemplo no pusiéramos la sentencia `contador++` , ya que la variable contador tendría siempre el mismo valor.