3.2.7 Bucles: compresión

Suponga que el análisis de las secuencias de temperaturas obtenidas desde un sensor indican que existe una gran redundancia en la información, pues es altamente probable que dos temperaturas consecutivas sean iguales. Se decide realizar un método simple de codificación para comprimer la información basado en el algoritmo *"run length encoding"* (*RLE*).

El algoritmo *RLE* se puede aplicar a secuencias de números reales —es más habitual en secuencias de enteros— en casos muy concretos donde sepamos que va a haber una gran redundancia. El concreto, consiste en codificar largas secuencias de valores idénticos consecutivos por un par de valores: el número que se repite más el número de repeticiones.

En nuestro caso, dado que el sensor realiza un muestreo muy denso en el tiempo, consideramos que es muy probable la repetición y, por tanto, decidimos realizar este tipo de codificación.

Ejercicio 3.9 Escriba un programa que realiza la codificación *RLE* para una secuencia de números reales que corresponden a temperaturas. Tenga en cuenta que los datos de entrada terminarán cuando se introduce una temperatura menor que el cero absoluto (−273.15 grados centígrados).

El resultado del programa será una secuencia de pares de la forma `<frecuencia><dato>` que termina en un último valor 0 que corresponde a un centinela para marcar que no hay más datos.

Algunos ejemplos de ejecución son:

Consola

```
1.9 1.9 1.9 1.9 2 2 2 2 2 2 2.1 2.1 2.1 2.2 -300
4 1.9 6 2 3 2.1 1 2.2 0
```

Consola

```
-300
0
```

donde la segunda línea corresponde a la salida del programa.

Lógicamente, para que la codificación sea útil será necesaria una aplicación que pueda obtener el fichero original.

Ejercicio 3.10 Escriba un programa que realiza la decodificación *RLE* para una secuencia de pares previamente codificados. Recuerde que cada par tiene la forma `<frecuencia><dato>` y que la secuencia termina con un valor 0 como centinela.

El resultado del programa es la secuencia original que podría haber producido esa codificación más un valor menor que el cero absoluto (−273.15) que marcaría el final de secuencia.

Además, el programa podrá terminar inesperadamente si detecta que la secuencia codificada muestra algún signo de ser errónea. Para ello, debe comprobar que cualquier valor de frecuencia es un entero positivo. Por ejemplo, si introducimos un grupo de datos sin codificar, es probable que haya valores negativos o valores con decimales que no pueden corresponder a una frecuencia.

Nota: Para comprobar si la frecuencia es errónea, lea el valor en una variable en coma flotante.

Algunos ejemplos de ejecución son:

Consola

```
4 1.9 6 2 3 2.1 1 2.2 0
1.9 1.9 1.9 1.9 2 2 2 2 2 2 2.1 2.1 2.1 2.2 -300
```

Consola

```
0
-300
```

donde la segunda línea corresponde a la salida del programa.

Sin olvidar, que tenemos que comprobar si hay un error. Por ejemplo:

Consola

```
4 1.9 6 2 3 2.1 1.5 2.2 0
1.9 1.9 1.9 1.9 2 2 2 2 2 2 2.1 2.1 2.1
Error inesperado, frecuencia no entera
```

Consola

```
4 1.9 6 2 3 2.1 -3 2.2 0
1.9 1.9 1.9 1.9 2 2 2 2 2 2 2.1 2.1 2.1
Error inesperado, valor de frecuencia sin sentido
```

donde puede ver que un valor de frecuencia menor que 1 no tiene sentido.