Ejercicio 9. *SOLUCIÓN*

Se muestran en rojo las sentencias erróneas y al lado, las sentencias corregidas:

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <math.h>
4
5   void Funcion1(float *media, float *varianza, float *desviacion, float *vector, int n);
6
7   void main(void)
8   {
9      int n, i;
10     float *datos, med, var, desv;
11
12     printf("Cuantos numeros va a introducir?  ");
13     scanf("%d", n);        ➔  scanf("%d", &n);
14     datos = malloc(n*sizeof(float));   ➔  datos = (float*)malloc(n*sizeof(float));
15     if(datos = NULL)   ➔  if(datos == NULL)
16       printf("\nError durante la reserva de memoria\n");
17     else
18     {
19        for(i=0; i<n; i++)
20        {
21           printf("Introduzca el dato %d: ", i+1);
22           scanf("%f", &(datos+i));   ➔  scanf("%f", datos+i);
23        }
24        Funcion1(med, var, desv, datos, n);   ➔  Funcion1(&med, &var, &desv, datos, n);
25        printf("\nLa media es %f\nLa varianza es %f\nLa desviacion es %f\n\n", med, var, desv);
26        free(datos);
27     }
28
29     system("PAUSE");
30  }
31
32  void Funcion1(float *media, float *varianza, float *desviacion, float *vector, int n)
33  {
34     int i;
35     *media = 0;
36     *varianza = 0;
37
38     for(i=0; i<=n; i++)   ➔  for(i=0; i<n; i++)
39        *media += *(vector+i);
40     *media /= n;
41
42     for(i=0; i<n; i++)
43        *varianza += (vector[i] - *media) * (vector[i] - *media);
44     *varianza /= n;
45     *desviacion = sqrt(*varianza);
46  }
```

NOTAS:

Funcion1 debe devolver tres variables como resultado (media, varianza y desviación). Para ello, recibe las variables *media*, *varianza* y *desviacion* mediante paso por dirección y almacena los resultados deseados en dichas variables (y no devuelve nada con *return*). Asimismo, también recibe como parámetro el vector que contiene las cantidades numéricas con que se trabaja. Cabe tener en cuenta que un vector siempre se pasa por dirección, por tanto, la declaración (línea 5):

```
void Funcion1(float *media, float *varianza, float *desviacion, float *vector, int n);
```

es correcta, como también lo hubiera sido la siguiente declaración:

```
void Funcion1(float *media, float *varianza, float *desviacion, float vector[], int n);
```

La línea 13 es incorrecta, porque el segundo parámetro de *scanf* debe ser la dirección de la celda de memoria donde se guarda el dato que el usuario introduce por teclado.

La línea 14 es incorrecta, porque la función *malloc* devuelve un puntero genérico que debe ser convertido a un puntero al tipo de datos con que se va a trabajar (en este caso, tipo *float*). Para ello, se debe poner *(float*)* delante de la llamada a *malloc*.

La línea 15 es incorrecta porque se está haciendo una asignación, cuando en realidad se desea hacer una comparación.

En la línea 22, a la función *scanf* le debemos indicar la dirección de la celda donde ir almacenando los números que introduce el usuario en cada iteración. Dado que se desea introducir en un vector, y el nombre del vector (*datos*) es la dirección de la celda 0, la dirección del resto de celdas se puede expresar como *datos+1*, *datos+2*, etc. De forma alternativa, también se pueden expresar las direcciones como *&datos[0]*, *&datos[1]*, etc. Por tanto, una forma alternativa de expresar esta sentencia es:

```
scanf("%f", &datos[i]);
```

La línea 24 es incorrecta, dado que los tres primeros parámetros se deben pasar por dirección. El cuarto parámetro es correcto, puesto que se trata de un vector. Un vector siempre se pasa por dirección, y el nombre del vector es la dirección del primer elemento, así que dicho parámetro es correcto.

Dentro de *Funcion1* es necesario inicializar el contenido de las variables *media* y *varianza* a cero, puesto que se trata de variables acumulador. También se podrían haber inicializado a 0 las variables *med* y *var* en *main*, y el resultado hubiera sido el mismo.

La línea 38 es incorrecta puesto que los índices de los elementos de un vector siempre corren desde 0 hasta el número de elementos menos uno.

La línea 39 es correcta, puesto que se está sumando en cada iteración el contenido de las celdas del vector. De forma alternativa, se podría haber utilizado notación vectorial tal y como sigue:

```
*media += vector[i];
```

La línea 43 es correcta. También se podría haber utilizado notación de puntero para acceder al contenido de las celdas, del siguiente modo:

```
*varianza += (*(vector+i) - *media) * (*(vector+i) - *media);
```