11.13. Soluciones a los ejercicios del capítulo

Listado 11.17: Ejercicio 11.1.

```
#include <stdio.h>

int main(int argc, char** argv)
{

        printf("Hola mundo, me llamo %s\n", argv[0]);

        return 0;
}
```

Listado 11.18: Ejercicio 11.2.

```
#include <stdio.h>

int main(int argc, char** argv)
{

        printf("Hola %s, soy el programa %s\n", argv[1], argv[0]);

        return 0;
}
```

Listado 11.19: Ejercicio 11.3.

```
#include <stdio.h>

int main(int argc, char** argv)
{

        printf("El nÃºmero de argumentos es %d\n", argc);

        return 0;
}
```

Listado 11.20: Ejercicio 11.4.

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv)
{
        float a, b, c;
```

```
7
8         a=atof(argv[1]);
9         b=atof(argv[2]);
10        c=a+b;
11
12        printf("%g+%g=%g\n", a, b, c);
13
14        return 0;
15 }
```

Listado 11.21: Ejercicio 11.5.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(int argc, char** argv)
5 {
6         int a, b, c;
7
8         a=atoi(argv[1]);
9         b=atoi(argv[2]);
10        /* Al ser a y b enteros, el resultado
11           será la parte entera de la división */
12        c=a/b;
13
14
15        printf("%d+%d=%d\n", a, b, c);
16
17        return 0;
18 }
```

Listado 11.22: Ejercicio 11.6.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(int argc, char** argv)
5 {
6         float a, b, c;
7
8         a=atof(argv[1]);
9         b=atof(argv[2]);
10
11        c=a-b;
12
13        if( c<0 ) {
14                c=-c;
```

```
15          }
16
17          printf("|%g-%g|=%g\n", a, b, c);
18
19          return 0;
20  }
```

Listado 11.23: Ejercicio 11.7.

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   int main(int argc, char** argv)
5   {
6           int i, n;
7           long int f;
8
9           n=atoi(argv[1]);
10
11          f=1;
12          for(i=2; i<=n; i++)
13                  f*=i;
14
15          printf("%d ! = %ld\n", n, f);
16
17          return 0;
18  }
```

Listado 11.24: Ejercicio 11.8.

```
1   #include <stdio.h>
2
3   int main(int argc, char** argv)
4   {
5           int n;
6           float f, df;
7
8           n=100;
9           df=1.0/n;
10
11          for(f=df; f<1.0; f=f+df)
12                  printf("%g\n", f);
13
14          return 0;
15  }
```

Listado 11.25: Ejercicio 11.9.

```
#include <stdio.h>

int main(int argc, char** argv)
{
        int n;

        n=2;
        do
        {
                printf("%d\n", n);
                n+=2;

        } while(n<=10);

        return 0;
}
```

Listado 11.26: Ejercicio 11.10.

```
#include <stdio.h>

int main(int argc, char** argv)
{
        int n;
        float f, df;

        n=100;
        df=1.0/n;

        f=df;
        while(f<1.0)
        {
                printf("%g\n", f);
                f=f+df;
        }

        return 0;
}
```

Listado 11.27: Ejercicio 11.11.

```
#include <stdio.h>

int main(int argc, char** argv)
{
        int i, j, k;
```

```
6         double A[3][3]={ {1.0, 2.0, 3.0},
7                          {2.0, 3.0, 1.0},
8                          {3.0, 1.0, 2.0} };
9         double B[3][3];
10        double C[3][3];
11
12        for(i=0; i<3; i++)
13        {
14                for(j=0; j<3; j++)
15                {
16                        printf("B[%d,%d]= ", i, j);
17                        scanf("%lf", &B[i][j]);
18                }
19        }
20
21        for(i=0; i<3; i++)
22        {
23                for(j=0; j<3; j++)
24                {
25                        C[i][j]=0.0;
26                        for(k=0; k<3; k++)
27                        {
28                                C[i][j]+=A[i][k]*B[k][j];
29                        }
30                }
31        }
32
33        for(i=0; i<3; i++)
34        {
35                for(j=0; j<3; j++)
36                {
37                        printf(" %4.4lf", C[i][j]);
38                }
39                printf("\n");
40        }
41
42        return 0;
43 }
```

Listado 11.28: Ejercicio 11.12.

```
1 #include <stdio.h>
2
3
4 int main(int argc, char** argv)
5 {
6         int npts, n;
7         double x, y;
```

```
8       double Sx, Sy, Sxy, Sxx, Syy;
9       double m_y, y0;
10      double m_x, x0;
11      double r2;
12      FILE *fin;
13
14      fin=fopen("datos.dat", "r");
15
16      npts=0;
17      Sx=0.0;
18      Sy=0.0;
19      Sxy=0.0;
20      Sxx=0.0;
21      Syy=0.0;
22
23      do
24      {
25              /* la función fscanf devuelve el número
26                 de datos leídos, que deben ser 2 */
27              n=fscanf(fin, "%lf%lf", &x, &y);
28              if( n==2 )
29              {
30                      npts++;
31                      Sx+=x;
32                      Sy+=y;
33                      Sxy+=x*y;
34                      Sxx+=x*x;
35                      Syy+=y*y;
36              }
37      } while( n==2 );
38
39      fclose(fin);
40
41      /* y=m_y*x+y0 */
42      m_y=(npts*Sxy-Sx*Sy)/(npts*Sxx-Sx*Sx);
43      y0 =(Sy-m_y*Sx)/npts;
44
45      /* x=m_x*y+x0 */
46      m_x=(npts*Sxy-Sx*Sy)/(npts*Syy-Sy*Sy);
47      /* y0 =(Sx-m_x*Sy)/npts; */
48
49      r2=m_x*m_y;
50
51      printf("Resultados de la regresion:\n");
52      printf("Y(X) = m_y * X + Y0 \n");
53      printf(" m_y= %g\n", m_y);
54      printf(" Y0 = %g\n", y0 );
55
```

```
56          printf("Coeficiente de correlacion r^2 = %g\n",
57                 r2);
58
59          return 0;
60  }
```

Listado 11.29: Ejercicio 11.13.

```
1   #include <stdio.h>
2
3   #define NMAXPTS 100
4
5   int leePuntos(char* arch, double x[], double y[])
6   {
7           int n, npts;
8           double xi, yi;
9
10          FILE *fin;
11
12          fin=fopen(arch, "r");
13
14          npts=0;
15          do
16          {
17                  n=fscanf(fin, "%lf%lf",
18                           &xi, &yi);
19                  if( n==2 && npts<NMAXPTS )
20                  {
21                          x[npts]=xi;
22                          y[npts]=yi;
23                  }
24                  npts++;
25
26          } while( n==2 && npts<=NMAXPTS );
27
28          fclose(fin);
29
30          /* error */
31          if( npts==NMAXPTS+1 && n==2 )
32          {
33                  npts=-1;
34          }
35
36          return npts;
37  }
38
39  double regresionLineal(int npts, double x[], double y[],
40                         double *_m, double *_y0)
```

```
{
        int n;
        double Sx, Sy, Sxy, Sxx, Syy;
        double m_x, m_y, y0;
        double r2;

        Sx=0.0;
        Sy=0.0;
        Sxy=0.0;
        Sxx=0.0;
        Syy=0.0;

        for(n=0; n<npts; n++)
        {
                Sx+=x[n];
                Sy+=y[n];
                Sxy+=x[n]*y[n];
                Sxx+=x[n]*x[n];
                Syy+=y[n]*y[n];
        }

        if( npts>0 )
        {
                /* y=m_y*x+y0 */
                m_y=(npts*Sxy-Sx*Sy)/(n*Sxx-Sx*Sx);
                y0 =(Sy-m_y*Sx)/npts;

                /* x=m_x*y+x0 */
                m_x=(npts*Sxy-Sx*Sy)/(n*Syy-Sy*Sy);

                r2=m_x*m_y;

                /* valores retornados */
                *_y0=y0;
                *_m =m_y;
        }
        else
        {
                r2=-1.0;
        }

        return r2;
}


int main(int argc, char** argv)
{
        int npts, n;
```

```
89          double x[NMAXPTS],
90                 y[NMAXPTS];
91          double m_y, y0, r2;
92
93          /* usa la función para leer los puntos */
94          npts=leePuntos("datos.dat", x, y);
95
96          /* manejar el error */
97          if( npts<0 )
98          {
99                 fprintf(stderr, "Error: el número de puntos"
100                       " excedió la memoria reservada\n");
101                exit(-1);
102         }
103
104         /* usa la función para calcular la regresión */
105         r2=regresionLineal(npts, x, y, &m_y, &y0);
106
107         /* manejar el error */
108         if( r2<0 )
109         {
110                fprintf(stderr, "Error: no se han pasado "
111                              "puntos a la función de "
112                              "regresión\n");
113                exit(-1);
114         }
115
116         printf("Resultados de la regresión:\n");
117         printf("Y(X) = m_y * X + Y0 \n");
118         printf(" m_y= %g\n", m_y);
119         printf(" Y0 = %g\n", y0 );
120
121         printf("Coeficiente de correlación r^2 = %g\n",
122                r2);
123
124         return 0;
125 }
```

Listado 11.30: Ejercicio 11.14.

```
1  #include <stdio.h>
2
3
4  void producto(double *p, double V[], double MxV[], double *modulo)
5  {
6      int i, j;
7
8      //calculamos el producto de la matriz y el vector
```

```
9
10      for (i=0; i<2; i++)
11          {
12                MxV[i]=0;
13                for (j=0; j<3; j++)
14                    MxV[i]+=p[3*i+j]*V[j];
15          }
16
17      //calculamos el módulo del vector resultante
18      for (i=0; i<2; i++)
19          *modulo+=MxV[i]*MxV[i];
20
21      *modulo=sqrt(*modulo);
22
23      return;
24  }
25
26
27  int main(int argc, char** argv)
28  {
29     double M[2][3]={{1.,1.,1.},{1.,1.,1.}};
30     double V[3]={1.,2.,3.};
31     double MxV[2], modulo;
32
33     producto(*M,V,MxV,&modulo);
34
35     printf("El producto de la matriz M por el vector V es (%g, %g)\n", MxV
           [0], MxV[1]);
36     printf("El modulo del vector es %f\n", modulo);
37
38     return 0;
39  }
```

Listado 11.31: Ejercicio 11.15.

```
1   #include <stdio.h>
2
3
4   void producto(double *p, double V[], double MxV[], double *modulo)
5   {
6       int i, j;
7
8       //calculamos el producto de la matriz y el vector
9
10      for (i=0; i<2; i++)
11          {
12                MxV[i]=0;
13                for (j=0; j<3; j++)
```

```
14                  MxV[i]+=p[3*i+j]*V[j];
15          }
16
17      //calculamos el módulo del vector resultante
18      for (i=0; i<2; i++)
19          *modulo+=MxV[i]*MxV[i];
20
21      *modulo=sqrt(*modulo);
22
23      return;
24  }
25
26
27  int main(int argc, char** argv)
28  {
29     double M[2][3]={{1.,1.,1.},{1.,1.,1.}};
30     double V[3]={1.,2.,3.};
31     double MxV[2], modulo;
32
33     producto(*M,V,MxV,&modulo);
34
35     printf("El producto de la matriz M por el vector V es (%g, %g)\n", MxV
           [0], MxV[1]);
36     printf("El modulo del vector es %f\n", modulo);
37
38     return 0;
39  }
```