

```

1 namespace DelegadosResumen
2 {
3     public class FunctionCall
4     {
5         public void CallingFirst()
6         {
7             Country country = new Country();
8             country.CreateCitiesList();
9
10            // Devuelve todos los componentes de la lista original.
11            List<City> listResult = country.Filter(() =>
12            {
13                return true;
14            });
15
16            // Otra sintaxis de igual resultado.
17            List<City> listResultAlt = country.Filter(() => true);
18
19            DisplayingResult(listResult);
20            Console.WriteLine();
21            DisplayingResult(listResultAlt);
22        }
23
24        public void CallingSecond()
25        {
26            Country country = new Country();
27            country.CreateCitiesList();
28
29            List<City> listResult = country.Filter((numberPop) =>
30            { return numberPop > 1000000; });
31
32            DisplayingResult(listResult);
33        }
34
35        public void CallingThird()
36        {
37            Country country = new Country();
38            country.CreateCitiesList();
39
40            List<City> listResult = country.Filter((numberPop) =>
41            { return (numberPop > 500000) && (numberPop <
42            1000000); });
43
44            DisplayingResult(listResult);
45        }
46
47        public void CallingFourth()
48        {
49            Country country = new Country();
50            country.CreateCitiesList();
51
52            List<City> listResult = country.Filter((name) => { return
53            name[0] == 'A'; });

```

```

54            DisplayingResult(listResult);
55        }
56
57        public void CallingFiveth()
58        {
59            Country country = new Country();
60            country.CreateCitiesList();
61
62            // Funciones anónimas.
63            Country.DelegateFilterPlain delegado = delegate () { return
64            true; };
65            List<City> listResult = country.Filter(delegado);
66
67            DisplayingResult(listResult);
68        }
69
70        public void CallingSixth()
71        {
72            Country country = new Country();
73            country.CreateCitiesList();
74
75            // sintaxis 1:
76            Country.DelegateSort comparator = delegate (City city1,
77            City city2)
78            {
79                if (city1.Population < city2.Population)
80                {
81                    return -1;
82                }
83                else
84                {
85                    return 1;
86                }
87            };
88            List<City> listResult = country.Sort(comparator);
89            DisplayingResult(listResult);
90
91            // sintaxis 2:
92            List<City> listResultALT = country.Sort((city1, city2) =>
93            {
94                if (city1.Population < city2.Population)
95                {
96                    return -1;
97                }
98                else
99                {
100                    return 1;
101                }
102            });
103            DisplayingResult(listResultALT);

```

```
101
102
103     public void DisplayingResult(List<City> list)
104     {
105         foreach (City c in list)
106         {
107             Console.WriteLine(c.Name + " with " + c.Population + " ➤
               population.");
108         }
109     }
110 }
111 }
```