```
namespace DelegadosResumen
{
    public class FunctionCall
    {
        public void CallingFirst()
        {
            Country country = new Country();
            country.CreateCitiesList();

            // Devuelve todos los componentes de la lista original.
            List<City> listResult = country.Filter(() =>
            {
                return true;
            });

            // Otra sintaxis de igual resultado.
            List<City> listResultAlt = country.Filter(() => true);

            DisplayingResult(listResult);
            Console.WriteLine();
            DisplayingResult(listResultAlt);
        }

        public void CallingSecond()
        {
            Country country = new Country();
            country.CreateCitiesList();

            List<City> listResult = country.Filter((numberPop) =>
              { return numberPop > 1000000; });

            DisplayingResult(listResult);
        }

        public void CallingThird()
        {
            Country country = new Country();
            country.CreateCitiesList();

            List<City> listResult = country.Filter((numberPop) =>
              { return (numberPop > 500000) && (numberPop <
              1000000); });

            DisplayingResult(listResult);
        }

        public void CallingFourth()
        {
            Country country = new Country();
            country.CreateCitiesList();

            List<City> listResult = country.Filter((name) => { return
              name[0] == 'A'; });
```



```

            DisplayingResult(listResult);
        }

        public void CallingFiveth()
        {
            Country country = new Country();
            country.CreateCitiesList();

            // Funciones anónimas.
            Country.DelegateFilterPlain delegado = delegate () { return
               true; };
            List<City> listResult = country.Filter(delegado);

            DisplayingResult(listResult);
        }

        public void CallingSixth()
        {
            Country country = new Country();
            country.CreateCitiesList();

            // sintaxis 1:
            Country.DelegateSort comparator = delegate (City city1,
              City city2)
                {
                    if (city1.Population < city2.Population)
                    {
                        return -1;
                    }
                    else
                    {
                        return 1;
                    }
                };
            List<City> listResult = country.Sort(comparator);
            DisplayingResult(listResult);


            // sintaxis 2:
            List<City> listResultALT = country.Sort((city1, city2) =>
            {
                if (city1.Population < city2.Population)
                {
                    return -1;
                }
                else
                {
                    return 1;
                }
            });
            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 }
```