Applicazione dei Delegati

I Delegati sono "riferimenti a funzioni" che possono essere assegnati ad apposite variabili.

Essi consentono di effettuare calcoli in modo flessibile poiché possono essere variabili non soltanto i dati su cui operare, ma anche la funzione di calcolo da utilizzare.

In pratica si ottiene l'effetto di poter memorizzare in variabili oltre ai valori anche le funzioni da utilizzare nei calcoli, **come se le funzioni a loro volta fossero dei dati da memorizzare.**

Per dimostrare l'utilità dei delegati, si propone una loro applicazione al calcolo di una espressione aritmetica senza parentesi.

La procedura di calcolo nella sua essenza è la seguente:

```
// per capire il metodo utilizzato si considera una semplice espressione con soltanto moltiplicazioni
string espressione = "12 * 5 * 4";
// innanzitutto metto gli elementi di una espressione in un array di oggetti
object[] arr = new object[5];
Func<int,int,int> m = Moltiplica;  // devo usare una variabile di appoggio
arr[0] = 12;
arr[1] = m;
arr[2] = 5;
arr[3] = m;
arr[4] = 4;

// ciclo sull'array
for(int i = 0; i<arr.Length; i++)
{
   if (arr[i] is Func<int,int,int>)
   {
      Func<int,int,int> f = arr[i] as Func<int,int,int>;
      int x = (int) arr[i-1];  // operando 1
      int y = (int) arr[i+1];  // operando 2
      // l'effettuazione del calcolo ricorda da vicino una notazione matematica
```

```
      int z = f(x, y);  // risultato parziale
      // lo metto nella casella i+1 per poterlo usare come prossimo operando
      arr[i+1] = z;
 }

// il risultato finale lo trovo nell'ultima casella dell'array
int risultatoFinale = (int) arr[arr.Length-1];

//////////////////////////////////////////

// funzioni di calcolo
public static int Moltiplica(int x, int y)
{ return x*y; }
public static int Dividi(int x, int y)
{ return x/y; }
public static int Somma(int x, int y)
{ return x+y; }
public static int Sottrai(int x, int y)
{ return x-y; }
```

Il programma completo è il seguente:

Form1.cs

```
using System;
using System.Windows.Forms;


namespace WindowsFormsApp2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }


// BOTTONE CHE FA ESEGUIRE IL CALCOLO
        private void button1_Click(object sender, EventArgs e)
        {
```

```
            string espressione = textBox1.Text;  // LEGGO DA UN TEXTBOX L'ESPR. DA CALCOLARE
            // aggiusto la forma dell'espressione se inizia con -
            if (espressione[0] == '-')
            { espressione = "0 " + espressione; }

            string[] appo = espressione.Split(' ');
            int numeroElementi = appo.Length;

            // l'array che contiene gli elementi dell'espressione
            object[] arr = new object[numeroElementi];
            // preparo i delegati
            Func<double, double, double> m = Moltiplica;
            Func<double, double, double> d = Dividi;
            Func<double, double, double> s = Somma;
            Func<double, double, double> t = Sottrai;

            // carico l'array con gli elementi dell'espressione
            for (int i = 0; i < numeroElementi; i++)
            {
                if (appo[i] == "*") { arr[i] = m; }
                else if (appo[i] == "/") { arr[i] = d; }
                else if (appo[i] == "+") { arr[i] = s; }
                else if (appo[i] == "-") { arr[i] = t; }
                else { arr[i] = Convert.ToDouble(appo[i]); }
            }

        /*
         per il calcolo si procede dapprima con tutte le * e / e poi con le + e -
         esempio di passaggi logici
         // innanzitutto calcolo * e /
            1 + 2 * 3 / 2 + 2 * 4
            1 +     6 / 2 + 2 * 4
            1 +         3 + 2 * 4
            1 +         3 +     8
         // ricompatto l'array
            1 + 3 + 8
         // infine calcolo + e -
            1 + 3 + 8
                4 + 8
                    12
         // il risultato rimane nell'ultima componente
         */


            // ciclo per calcolare tutte le moltiplicazioni e le divisioni
            // con associatività a sinistra
            for (int i = 1; i < arr.Length - 1; i++)
            {
                if (arr[i] is Func<double, double, double>) // se è un operatore
                {
                    // casting
                    Func<double, double, double> f = arr[i] as Func<double, double, double>;
                    if (f == m || f == d)  // moltiplicazione o divisione
                    {
                        double x = (double)arr[i - 1];
                        double y = (double)arr[i + 1];
                        double risultato = f(x, y);
                        arr[i - 1] = null;
                        arr[i] = null;
                        arr[i + 1] = risultato;
                        numeroElementi = numeroElementi - 2;
                    }
                }
            }
```

```
            // rimuovo le caselle null
            // uso un array di appoggio
            object[] arr2 = new object[numeroElementi];
            int k = 0;
            for (int i = 0; i < arr.Length; i++)
            {
                if (arr[i] != null)
                { arr2[k] = arr[i]; k++; }
            }
            // riassegno arr2 all'array di partenza arr
            arr = arr2;

            // ciclo per calcolare tutte le somme e le sottrazioni
            // con associatività a sinistra
            for (int i = 1; i < arr.Length - 1; i++)
            {
                if (arr[i] is Func<double, double, double>) // se è un operatore
                {
                    // casting
                    Func<double, double, double> f = arr[i] as Func<double, double, double>;
                    if (f == s || f == t)  // somma o sottrazione
                    {
                        double x = (double)arr[i - 1];
                        double y = (double)arr[i + 1];
                        double risultato = f(x, y);
                        arr[i - 1] = null;
                        arr[i] = null;
                        arr[i + 1] = risultato;
                    }
                }
             }

            // il risultato finale è nell'ultimo elemento dell'array
            label1.Text = arr[numeroElementi - 1].ToString();  // LO SCRIVO IN UNA LABEL
        }


        // funzioni di calcolo
        public static double Moltiplica(double x, double y)
        { return x * y; }

        public static double Dividi(double x, double y)
        { return x / y;  }

        public static double Somma(double x, double y)
        { return x + y; }

        public static double Sottrai(double x, double y)
        { return x - y; }

    }
}
```