Делегати

Делегат (delegate) је тип података који омогућава смештање референци на функције.
Делегати се декларишу слично функцијама, али немају тело као функција и користе службену реч delegate.
Декларација делегата специфицира повратни тип и листу параметара.
После дефинисања делегата, може се декларисати промењива типа као и делегат.
Затим се може иницијализовати промењива као референца према било којој функцији која има исти повратни тип и листу параметара као и делегат.
Када се то заврши, може се позвати функција коришћењем делегат промењиве као да је у питању функција.

Када постоји промењива која реферише на функцију, могу се реализовати и друге операције које би иначе било немогуће реализовати.
Нпр, може се додати делегат промењива функцији као параметар а затим та функција може да користи делегата за позив на било коју функцију на коју има референцу, без знања о којој функцији је реч током извршења програма.
Пример:

```
using System;
namespace Proba
{
    class Program
    {
        delegate double ObradaDelegata(double x, double y);
        static double Mnozenje(double x, double y) => x * y;
        static double Deljenje(double x, double y) => x / y;
        static void Main(string[] args)
        {
            ObradaDelegata obrada;
            Console.WriteLine("Uneti dva broja odvojena zarezom:");
            string ulaz = Console.ReadLine();
            int pozicijaZareza = ulaz.IndexOf(',');
            double x = Convert.ToDouble(ulaz.Substring(0,
                    pozicijaZareza));
            double y = Convert.ToDouble(ulaz.Substring(pozicijaZareza
                    + 1, ulaz.Length - pozicijaZareza - 1));
            Console.WriteLine("Uneti M za mnozenje ili D za
                    deljenje:");
            ulaz = Console.ReadLine();
            if ((ulaz == "M") || (ulaz == "m"))
                obrada = new ObradaDelegata(Mnozenje);
            else
                obrada = new ObradaDelegata(Deljenje);
            Console.WriteLine($"Rezultat: {obrada(x, y)}");
        }
    }}
```

Даје:
Uneti dva broja odvojena zarezom:
2.05, 3.35
Uneti M za mnozenje ili D za deljenje:
m
Rezultat: 6.8675

Код дефинише делегата (`ObradaDelegata`) чији повратни тип и параметри одговарају онима из две функције: `Mnozenje(), Deljenje()` и обе функције користе => ламбда стрелицу, методу тела израза.
Службена реч delegate специфицира да је дефиниција за делегата а не за функцију.
Дефиниција специфицира и да је double повратна вредност као и два double параметра.
Имена која се користе нису битна нити за тип делегата нити за имена параметара.
У примеру се користе име делегата `ObradaDelegata` и параметри названи x и y.
У главном делу кода се декларише промењива као нов тип делегата: `ObradaDelegata obrada;` .
Затим се уносе две вредности које се одвајају зарезом, па се корисник пита за жељену операцију над тим вредностима.

Да би се доделила референца функције на делегат промењиву, синтакса је налик додели низовних вредности пошто се користи new службена реч за креирање новог делегата.

```
if ((ulaz == "M") || (ulaz == "m"))
    obrada = new ObradaDelegata(Mnozenje);
else
    obrada = new ObradaDelegata(Deljenje);
```

После new се специфицира тип делегата и даје аргумент који прави референцу на функцију која се жели користити (`Mnozenje` или `Deljenje`).

Овај аргумент не одговара параметрима типа делегата или циљаној функцији; ово је синтакса јединствена за доделу делегата.
Аргумент је просто име функције која се жели користити без заграда.
Постоји и једноставнија синтакса:

```
if ((ulaz == "M") || (ulaz == "m"))
    obrada = Mnozenje;
else
    obrada = Deljenje;
```

Компајлер препознаје да тип делегата промењиве промењиве која се обрађује одговара потпису две функције и аутоматски иницијализује жељеног делегата.
На крају се позива жељена функција коришћењем делегата без обзира на коју се функцију делегат позива:

```
Console.WriteLine($"Rezultat: {obrada(x, y)}");
```

Овде се делегат промењива третира као да је име функције.

За разлику од функције, ипак, може се извести додатне операције на овој промењивој, попут додавања промењиве према функцији преко параметара:
static void IzvrsenjeFunkcije(ObradaDelegata obrada) => obrada(2.2, 3.3)
Ово значи да се може контролисати понашање функције додавањем функција делегата.
Нпр, можа постоји функција која сортира низ по алфабету.
Постоје многе технике за сортирање листе, али коришћењем делегата, може се специфицирати функција која се користи додавањем сортирајућег алгоритма делегата функције у функцију за сортирање.