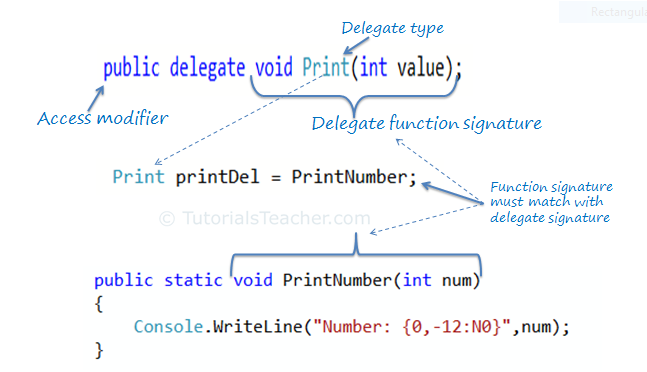
**Delegates** (https://www.tutorialsteacher.com/csharp/csharp-delegates)

→ wat ?

Functie als parameter doorgeven

Call back function of event handler

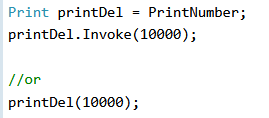
* Delegate (pointer naar een functie)



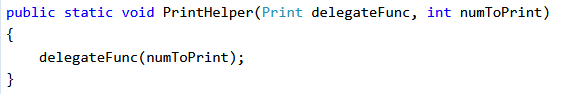
→ Alternatieve initialisatie :



→ Invoke delegate (functie oproepen)

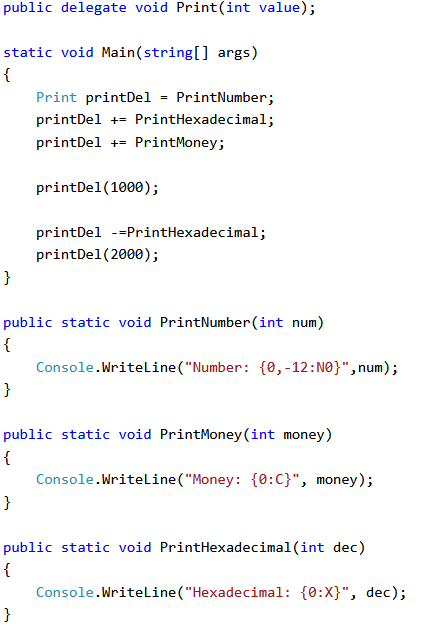


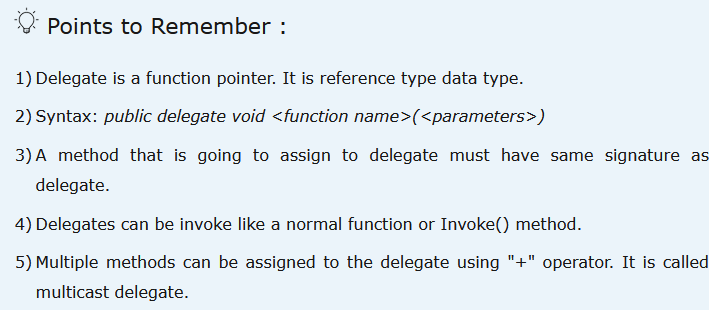
→ delegate als parameter



→ multicast delegate

Een delegate kan naar meerdere methodes/functies wijzen (=multicast delegate). Om functies oe te voegen kan “+” gebruikt worden om methodes te verwijderen “-“.



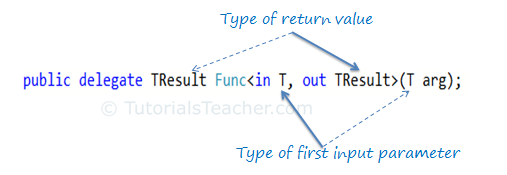


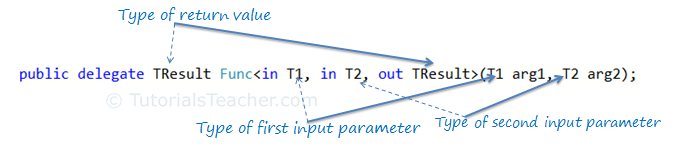
**Func /Action**

(<https://www.tutorialsteacher.com/csharp/csharp-func-delegate>)

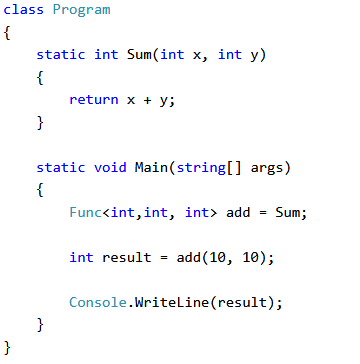
(<https://www.tutorialsteacher.com/csharp/csharp-action-delegate>)

Func is een generieke delegate in de System namespace en heeft nul of meerdere input parameters en 1 output parameter. De laatste parameter wordt beschouwd als de output.

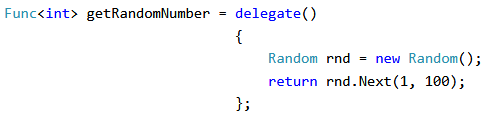




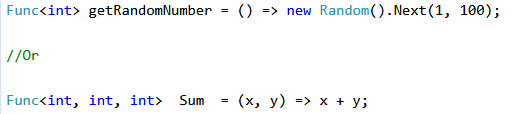
Voorbeeld :

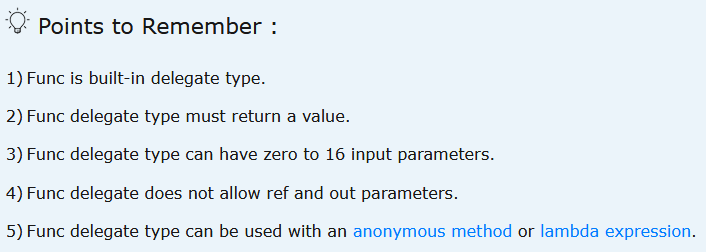


Func in combinatie met een anonieme methode :



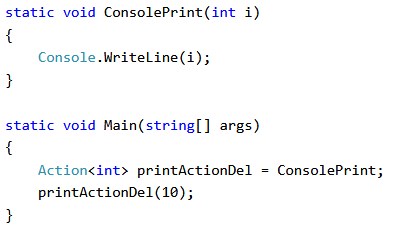
Func in combinatie met lambda expressie :



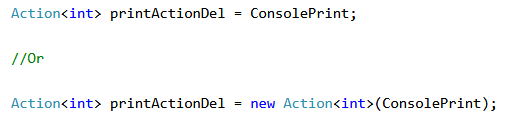


Action is eveneens een delegate type gedefinieerd in de System namespace, maar geeft in tegenstelling tot Func geen waarde terug. (vergelijk met methode dat void als return type heeft)

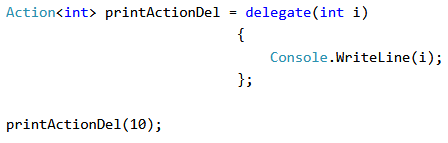
Voorbeeld :

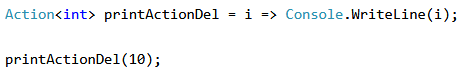


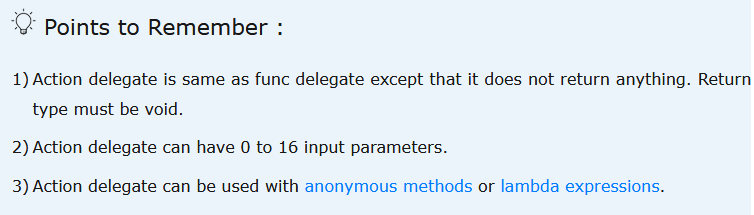
Initialisatie kan door direct een functie toe te kennen of door middel van new.



Ook hier kan gebruik gemaakt worden van anonieme functies of lambda expressies :





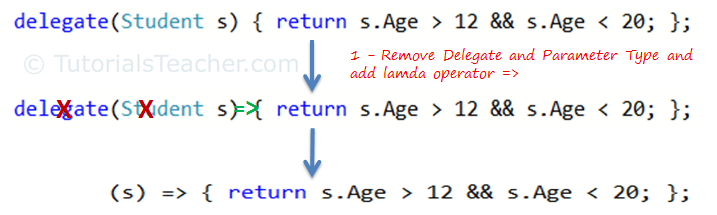


**Lambda functies**

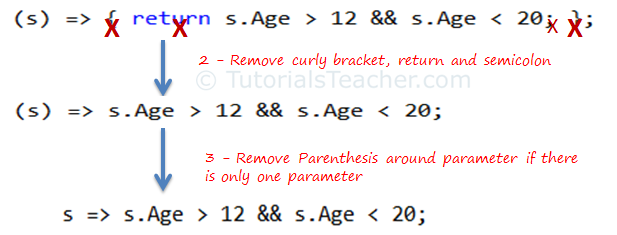
(<https://www.tutorialsteacher.com/linq/linq-lambda-expression>)

Is een kortere manier om anonieme methodes te schrijven.

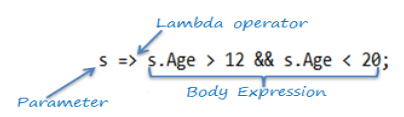
Voorbeeld overgang van anonieme methode naar lambda expressie :



Hebben we te maken met een functie die uit slechts 1 statement bestaat en een waarde terug geeft dan kunnen we verder vereenvoudigen tot :



Syntax :



Lambda expressie zonder parameters :



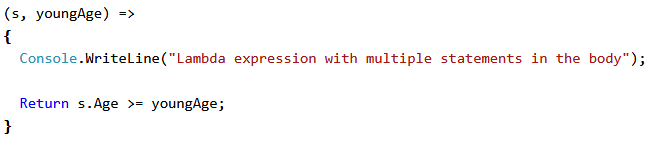
Lambda expressie met meerdere parameters :

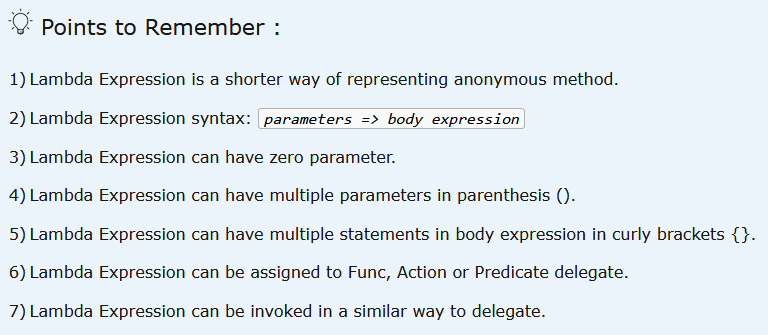


Of



Lambda expressies kunnen ook meerder statements bevatten :



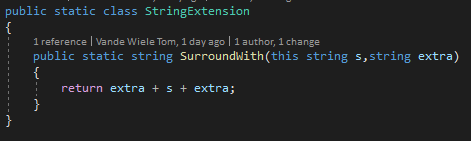


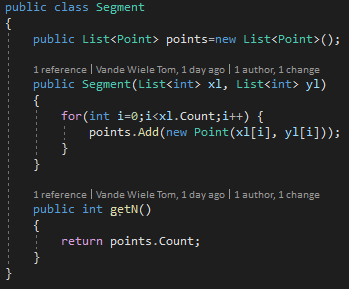
**Extension method**

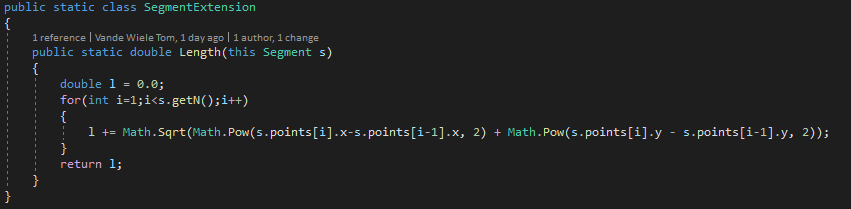
(<https://www.tutorialsteacher.com/csharp/csharp-extension-method>)

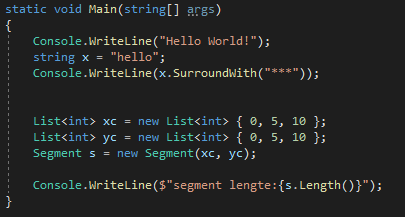
Extension methods laten toe om extra methodes toe te voegen aan classen zonder de originele classen aan te passen of overerving te gebruiken.

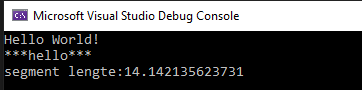
Voorbeeld :

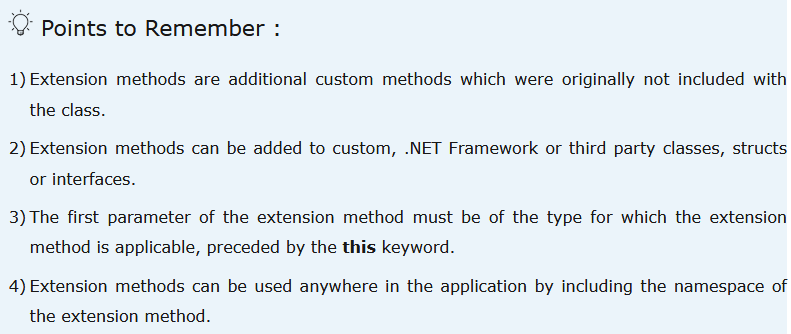












**Events**

(<https://www.tutorialsteacher.com/csharp/csharp-event>)

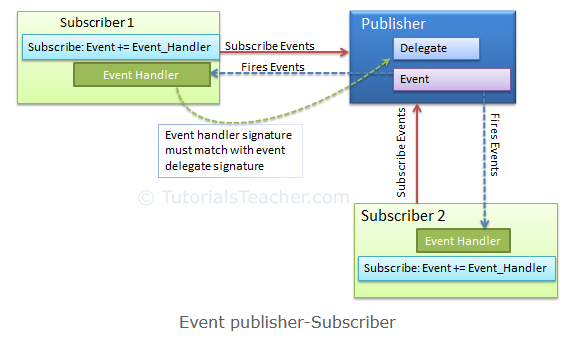
(<https://docs.microsoft.com/en-us/dotnet/standard/events/>)

(<https://nl.wikipedia.org/wiki/Observer_(ontwerppatroon)>)

Een event is een boodschap die door een object wordt gestuurd om te laten weten dat een bepaalde actie heeft plaats gevonden. Het object dat het event stuurt noemen we de ‘event sender’, dit object weet niet naar wie de boodschap wordt verstuurd.

Het design pattern dat hierbij aansluit is het ‘observer design pattern’. We hebben hierbij een ‘publisher’, het object dat de boodschap stuurt en 1 of meerdere ontvangers of ‘subscribers’. Deze ontvangers kunnen zich registreren bij de ‘publisher’ om de desgewenste boodschappen (notifications) te ontvangen.

Dit patroon wordt vooral gebruikt om een zwakke koppeling tussen objecten te realiseren, zodat objecten vrij onafhankelijk van elkaar kunnen bestaan.



Voorbeeld

