# Informatica: C# WPO 11

## 1. Inhoud

Pass by reference (ref, out), recursie, code opdelen in verschillende codebestanden

## 2. Oefeningen

- Demo 1: Swapfunctie
- Demo 2: TryParse(int)
- Demo 3: Recursion Tree
- Demo 4: DrawingLibrary
- A: TryParse(double)
- A: Solve equation
- A: Generic equation solver
- A: Klavertje vier
- A: Bubble sort
- E: Binary search

### 2.1 Demo 1: Swapfunctie

In dit programma wordt een swapfunctie geschreven die toelaat om 2 getallen van klein naar groot te sorteren indien nodig. Laat deze los op een array van ongesorteerde getallen. Print zowel de originele array als het resultaat na sorteren af. De swapfunctie wordt hierbij opgeroepen met 2 argumenten die "by reference" worden doorgegeven.

### 2.2 Demo 2: TryParse(int)

De klassieke "int.Parse" conversie laat toe om string naar een geheel getal om te zetten. Echter kan het zijn dat een gebruiker een gewoon karakter ingeeft waardoor deze omzettingsmethode niet goed zal functioneren. Schrijf een functie die eerst nagaat of de ingegeven tekst effectief enkel een getal bevat. Indien dit zo is wordt deze tekst omgezet in een getal. De functie retourneert een boolean waarde (true indien gelukt, false indien niet gelukt). De waarde na omzetting (integer) wordt via de "out"-statement aan de oproepende functie teruggestuurd.

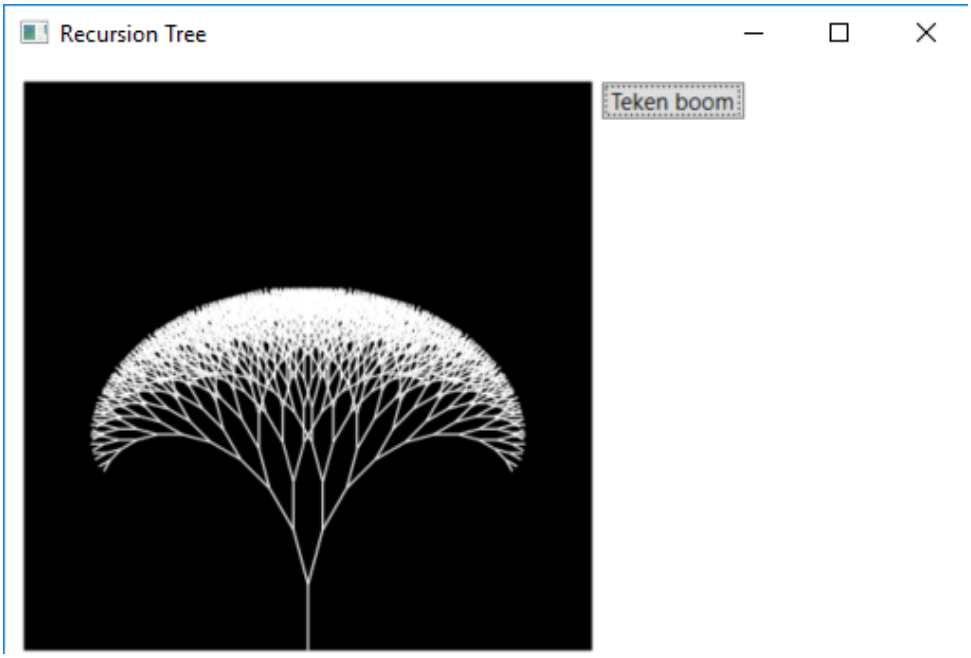
## 2.3 Demo 3: Recursion Tree

In deze demo zal een eenvoudig voorbeeld van recursie getoond worden a.d.h.v. de recursieboom (zie afbeelding 1). Deze recursieboom wordt opgebouwd uit een tak die zich opsplitst in 2 subtakken. Door deze procedure te herhalen bekomt men de recursieboom. De parameters die ervoor zorgen dat men een mooie boom kan bekomen zijn als volgt (neem $P_1$ als het huidige aangrijpingspunt van de tak):

$$\begin{cases} l_a = x \cdot l_{a-1} \\ \alpha_a = \alpha_{a-1} + \theta \\ \alpha_b = \alpha_{a-1} - \theta \end{cases} \tag{1}$$

$$\begin{cases} P_x = P_{x-1} + l_{a-1} \cdot cos(\alpha_{a-1}) \\ P_y = P_{y-1} + l_{a-1} \cdot sin(\alpha_{a-1}) \end{cases} \tag{2}$$

Waarbij $x$ de verhouding is de huidige lengte $l_{a-1}$ en de lengte van de 2 subtakken $l_a$, $\alpha$ de hoek van de tak is, en $P$ de aangrijpingspunten zijn van de lijnstukken. In deze opgave mag je $x$ en $\theta$ respectievelijk gelijkstellen aan 15 (graden) en 0.85.




**Figuur** 1: Recursieboom.

## 2.4 Demo 4: DrawingLibrary

In de meeste van je toepassingen zal je iets op een canvas tekenen. Dit tekenen herleidt zich steeds hetzelfde en eigenlijk is het een goede praktijd als je dezelfde code in verschillende projecten kan hergebruiken. In deze demo zullen we een aantal tekenfuncties in een aparte bibliotheek onderbrengen die we dan in verschillende projecten kunnen overnemen. Hier wordt het gebruik van nieuwe klassen (met static methoden) aangetoond.

## 2.5 A: TryParse(double)

De klassieke "double.Parse" conversie laat toe om string naar een kommagetal om te zetten. Echter kan het zijn dat een gebruiker een gewoon karakter ingeeft (anders dan de punt of komma) waardoor deze omzettingsmethode niet goed zal functioneren. Schrijf een functie die eerst nagaat of de ingegeven tekst effectief enkel een getal bevat. Indien dit zo is wordt deze tekst omgezet in een getal. De functie retourneert een boolean waarde (true indien gelukt, false indien niet gelukt). De waarde na omzetting (double) wordt via de "out"-statement aan de oproepende functie teruggestuurd.

## 2.6 A: Solve equation

In deze opdracht ga je een eenvoudig rekenmachine maken die een vergdelijking kan inlezen en het resultaat ervan kan retourneren. Als resultaat mag je steeds uitgaan van het type double. De vergelijkingen die je gaat uitvoeren zien er als volgt uit:

- 3+4
- 100*52
- 52/12
- 54-52.6

De vergelijkingen bestaan dus steeds uit 2 getallen gescheiden door een operator. Schrijf 4 verschillende functies die toelaten om 1 formaat van vergelijking op te lossen (vermenigvuldiging, deling, optelling en subtractie). Elk van deze functies gaat een stringsplit operatie op de vergelijking (formaat string). Heb je 2 elementen in de array van gesplitste waarden, dan kan je de operatie uitvoeren, anders niet. Een functie-prototype kan er als volgt uitzien: "private bool multiply(string vgl, out double result)".

## 2.7 A: Generic equation solver

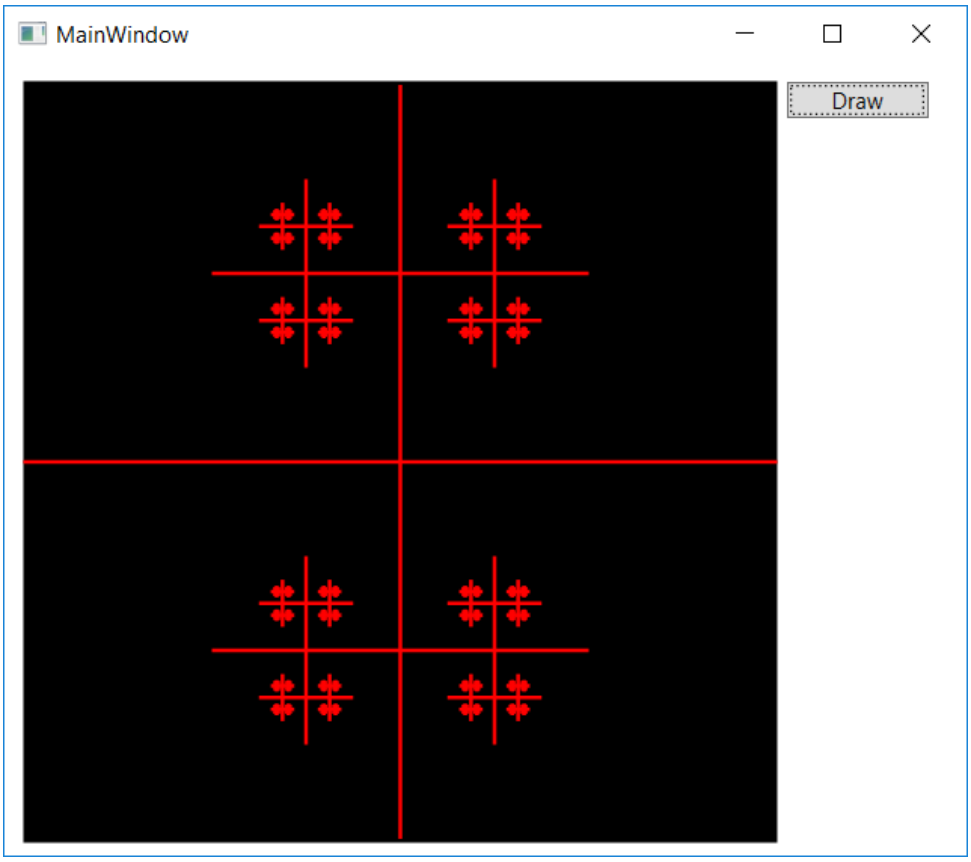
Herneem de voorgaande opgave. Kan je de 4 functies onderbrengen in een aparte bibliotheek? In de voorgaande opgave heb je waarschijnlijk een enkele functie uitgeprobeerd op een gegeven input van de gebruiker. Kan je er ook voor zorgen dat je een $5^{de}$ functie schrijft die de 4 andere functies uitprobeert totdat de vergelijking opgelost is? Indien geen enkele van de 4 andere functies een succes retourneert, retourneert deze functie ook een negatief resultaat. De functie-prototype van deze $5^{de}$ functie is als volgt: "public static bool tryEquation(string vgl, out double result)".

## 2.8 A: Klavertje vier

Teken een klavertje 4 zoals afgebeeld in figuur 2. Deze recursieve tekening werkt als volgt:

- Teken een horizontale lijn die de hele breedte van de canvas inneemt.

- Deel de breedte door 2 (=lengte) en laat 2 lijnen vertricaal vertrekken vanuit het middelste van de eerste lijn. De 2 nieuwe lijnen gaan naar boven en naar beneden vanuit dit middelste punt. De lengte van 2 nieuwe lijnen is half zo lang t.o.v. de eerste lijn.
- Voor elk van de nieuwe lijnen teken je opnieuw 2 nieuwe lijnen die op 90 graden georiënteerd staan t.o.v. de oude lijn. De lijnen zijn telkens half zo lang als de oude lijn en vertrekken steeds vanuit het midden van de oude lijn.
- Na 8 tot 10 recursiedieptes bekom je de klaver zoals weergegeven in de afbeelding. Zorg ervoor dat de recursie kan staoppen na een bepaalde diepte.




**Figuur** 2: Klavertje vier.

## 2.9 A: Bubble sort

Bubble sort is een bekend en eenvoudig algoritme om een array met getallen van klein naar groot (of omgekeerd) te sorteren. Schrijf het algoritme en maak gebruik van de swap-functie om 2 naburige getallen van volgorde te wisselen. Maak eerst een randomarray aan die je door buuble-sort-algoritme laat gaan. Merk op dat een array doorgeven als argument eigenlijk steeds by reference wordt doorgestuurd. Hierdoor volstaat volgende functie-prototype: “public static void bubblesort(int[] values)”.

## 2.10 E: Binary search

Herneem de oplossing van de bubble sort. Binary search is een zoekalgoritme die toelaat om in een zeer snel tempo een waarde (of naburige) waarden in een array te zoeken. Hierbij dient de array eerst gesorteerd te zijn van klein naar groot. Binary search is een recursief zoekalgoritme die als volgt werkt:

1. Kies eerst een waarde waarop je wilt zoeken.
2. Neem de totale lengte van de array en deel die door 2. Dit getal wordt onze nieuwe index.
3. Vergelijk de array op deze plaats (zoekindex) met de waarde uit puntje 1. Zijn beide gelijk, dan retourneer je een booleaanse true en geef je de index (positie in array) van de gevonden waarde terug van een waarde by reference. Indien dit niet het geval is, zoeken we verder.
4. Is de waarde uit de array kleiner dan de zoekwaarde, dan neem je de linkerhelft van de array ([0 tot n/2]). Deze halve aray wordt de nieuwe zoekarray. Ga recursief door naar puntje 2.
5. In het andere geval neem je de rechterhelft ([n/2 tot n-1]). Deze halve aray wordt de nieuwe zoekarray. Ga recursief door naar puntje 2.
6. Is het aantal elementen in je array kleiner dan 2 en heb je de gewenste waarde niet gevonden? In dat geval retourneer je recursief false terug en geef je als waarde van de index (by reference) een -1 terug.