

Opdracht 'Faculteit'

1!	= 1	= 1	1! = 1
2!	= 1 x 2	= 2	2! = 2 x 1!
3!	= 1 x 2 x 3	= 6	3! = 3 x 2!
...			
9!	= 1 x 2 x ... x 8 x 9	= 362.880	9! = 9 x 8!
10!	= 1 x 2 x ... x 9 x 10	= 3.628.800	10! = 10 x 9!
n!	= 1 x 2 x ... x (n-1) x n	= n!	n! = n x (n-1)!

Opdracht 16A (Console Applicatie) – 1 punt

Lees een getal in (n). Bepaal en toon de faculteit van het getal (n!) m.b.v. methode 'Faculteit'. Maak geen gebruik van recursie, maar van **iteratie** (loop). Zorg voor onderstaande uitvoer.

```

file:///C:/Users/gerwin/documents/visual st...
n = 7
7! = 1 x 2 x 3 x 4 x 5 x 6 x 7 = 5040
  
```

Opdracht 16B (Console Applicatie) – 2 punten

Maak opdracht 16A nogmaals, maar maak nu gebruik van een **recursieve** methode 'Faculteit'. We kunnen gebruik maken van de recurrente betrekking hiernaast.

Als een uitvoer zoals linksonder getoond wordt, dan 1 punt.

Als een uitvoer zoals rechtsonder getoond wordt, dan 2 punten.

$$\begin{cases} f_n = f_{(n-1)} \times n & n > 1 \\ f_1 = 1 \end{cases}$$

Voorwaardelijke eisen:

- De recurrente betrekking (formule) moet duidelijk te herkennen zijn in de code.
- De recursieve methode wordt 1x van buitenaf (dus vanuit de Main) aangeroepen.
- De recursieve methode retourneert het (eind)antwoord naar de Main-methode.
- Het printen moet binnen de recursieve methode gebeuren.

```

file:///C:/Users/gerwin/documents/vi...
n = 7
f(n) = 1 x 2 x 3 x 4 x 5 x 6 x 7 = 5040
  
```

1

```

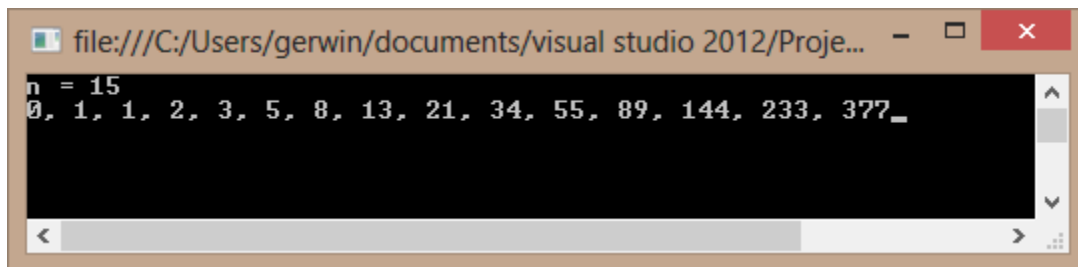
file:///C:/Users/gerw...
n = 7
f(1) = 1
f(2) = 2 x f(1) = 2
f(3) = 3 x f(2) = 6
f(4) = 4 x f(3) = 24
f(5) = 5 x f(4) = 120
f(6) = 6 x f(5) = 720
f(7) = 7 x f(6) = 5040
  
```

Opdracht 'Fibonacci'

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

Opdracht 17A (Console Applicatie) – 1 punt

Lees een getal in (n). Bepaal en toon de eerste n getallen van de (Fibonacci) reeks m.b.v. methode 'Fibonacci'. Maak geen gebruik van recursie, maar van **iteratie** (loop). Laat 2 variabelen (getal1 en getal2) 'meelopen' met de reeks. Zorg voor onderstaande uitvoer.



```
file:///C:/Users/gerwin/documents/visual studio 2012/Proje...
n = 15
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377_
```

Opdracht 17B (Console Applicatie) – 2 punten

Maak opdracht 17A nogmaals, maar maak nu gebruik van **recursieve** methode 'Fibonacci'. We kunnen gebruik maken van de hiernaast beschreven recurrente betrekking.

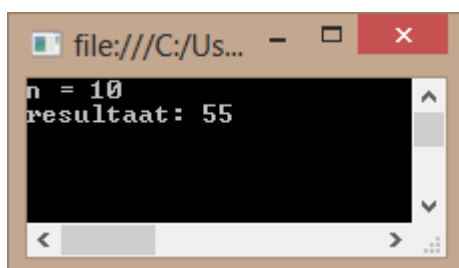
Als een uitvoer zoals linksonder getoond wordt, dan 1 punt.

Als een uitvoer zoals rechtsonder getoond wordt, dan 2 punten.

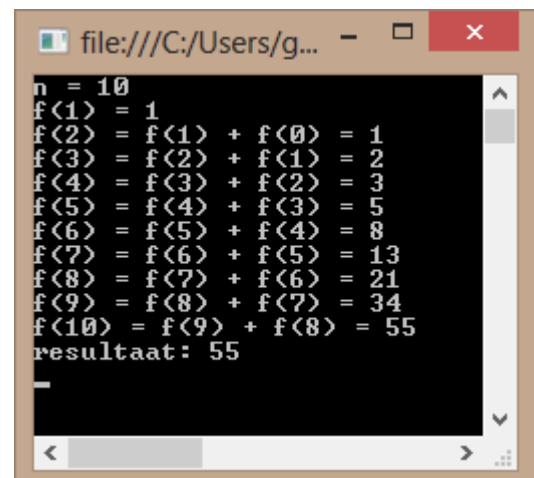
$$\begin{cases} f_n = f_{(n-1)} + f_{(n-2)}, & n \geq 2 \\ f_0 = 0 \\ f_1 = 1 \end{cases}$$

Voorwaardelijke eisen:

- De recurrente betrekking (formule) moet duidelijk te herkennen zijn in de code.
- De recursieve methode wordt 1x van buitenaf (dus vanuit de Main) aangeroepen.
- De recursieve methode retourneert het (eind)antwoord naar de Main-methode.
- Het printen moet binnen de recursieve methode gebeuren.



```
file:///C:/Us...
n = 10
resultaat: 55
```



```
file:///C:/Users/g...
n = 10
f<1> = 1
f<2> = f<1> + f<0> = 1
f<3> = f<2> + f<1> = 2
f<4> = f<3> + f<2> = 3
f<5> = f<4> + f<3> = 5
f<6> = f<5> + f<4> = 8
f<7> = f<6> + f<5> = 13
f<8> = f<7> + f<6> = 21
f<9> = f<8> + f<7> = 34
f<10> = f<9> + f<8> = 55
resultaat: 55
```

Opdracht Triangles

2 punten

Teken het 'driehoeken'-patroon dat op eerste slide staat afgebeeld van de presentatie van 'Wiskunde Theorie' (les 1) m.b.v. van een recursieve methode `DrawTriangle`. Gebruik een Windows Forms applicatie waarin je het event `Form1_Paint` kunt gebruiken om de recursieve methode `DrawTriangle` aan te roepen.

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    DrawTriangle(e.Graphics, ???, ???, ...);
}
```

Houd rekening met het volgende:

- de recursieve methode `DrawTriangle` moet weten waar de driehoek getekend moet worden; dat kan met (bv): hoogste punt (x, y) en breedte en hoogte; aan de hand van deze informatie kun je de drie hoekpunten bepalen, en lijnen tussen deze 3 hoekpunten tekenen;
- gebruik voor het tekenen van een lijn '`Graphics.DrawLine(Pen pen, int x1, int y1, int x2, int y2)`';
- voor het tekenen van de sub-driehoeken roep je `DrawTriangle` weer aan, maar dan met aangepaste parameters; hoeveel sub-driehoeken moet je steeds tekenen?
- zorg voor een stopconditie, zodat de recursieve methode niet eindeloos wordt aangeroepen;

