In [17]:
```
import numpy as np
from IPython.display import Image
Image('pictures/1_1.png', height=300)
```

Out[17]:

1.1 Ciągi liczbowe

Dla następujących ciągów liczbowych

1. $\forall n \in \{1, 2, \dots\} : x(n) = 3^n + x(n-1)$
   $x(0) = 0$
2. $\forall n \in \{1, 2, \dots\} : x(n) = n + x(n-2)$
   $x(-1) = x(0) = 0$
3. $\forall n \in \{2, 3 \dots\} : x(n) = x(n-1) + x(n-2)$
   $x(1) = 1; x(0) = 0$

wykonaj:

1. zaimplementuj (rekurencyjny) algorytm wyliczający wartość $n$tego elementu ciągu,
2. analitycznie wyznacz wzór na wartość $n$tego elementu ciągu (indukcja),
3. napisz procedurę weryfikującą poprawność zaimplementowanej rekurencji (wyświetlającą i porównującą wynik numeryczny i analityczny) dla N pierwszych elementów ciągu.

Podpunkt 1.

Rekurencyjny algorytm na n-ty wyraz ciągu:

$$\forall n \in \{1, 2, \dots\} : x(n) = 3^n + x(n-1)$$
$$x(0) = 0$$

Analityczny wzór:

$$x(n) = \frac{3}{2}(3^n - 1)$$

In [2]:
```
# punkt 1
def seq1(n):
    if n == 0:
        return 0
    else:
        return 3**n + seq1(n-1)

def seq1_analitic(n):
    return 3/2 * (3 ** n -1)



for i in range(10):
    print('i =', i)
    print('algorytm x(i) =', seq1(i))
    print('analitycznie x(i) =', seq1_analitic(i))
```

```
i = 0
algorytm x(i) = 0
analitycznie x(i) = 0.0
i = 1
algorytm x(i) = 3
analitycznie x(i) = 3.0
i = 2
algorytm x(i) = 12
analitycznie x(i) = 12.0
i = 3
algorytm x(i) = 39
analitycznie x(i) = 39.0
i = 4
algorytm x(i) = 120
analitycznie x(i) = 120.0
i = 5
algorytm x(i) = 363
analitycznie x(i) = 363.0
i = 6
algorytm x(i) = 1092
analitycznie x(i) = 1092.0
i = 7
algorytm x(i) = 3279
analitycznie x(i) = 3279.0
i = 8
algorytm x(i) = 9840
analitycznie x(i) = 9840.0
i = 9
algorytm x(i) = 29523
analitycznie x(i) = 29523.0
```

Podpunkt 2

Rekurencyjny zapis:

$$\forall n \in \{1, 2, \dots\} : x(n) = n + x(n-2)$$
$$x(1) = 1; x(0) = 0$$

wzór analityczny:

$$x(n) = \frac{1}{8}(2(n+1)(n+2) + (-1)^{n+1} + (-1)^{2n+1}(2n+3))$$

In [22]:
```
# punkt 2
def seq2(n):
    if n == 0 or n == -1:
        return 0
    else:
        return n + seq2(n-2)

def seq2_anltc(n):
    return 1/8*(2*(n+1)*(n+2)+(-1)**(n+1)+(-1)**(2*n+1)*(2*n+3))

for i in np.random.randint(0, 200):
    if seq2(i) != seq2_anltc(i):
        print(AssertionError)

for i in range(10):
    print('i =', i)
    print('algorytm x(i) =', seq2(i))
    print('analitycznie x(i) =', seq2_anltc(i))
```

```
i = 0
algorytm x(i) = 0
analitycznie x(i) = 0.0
i = 1
algorytm x(i) = 1
analitycznie x(i) = 1.0
i = 2
algorytm x(i) = 2
analitycznie x(i) = 2.0
i = 3
algorytm x(i) = 4
analitycznie x(i) = 4.0
i = 4
algorytm x(i) = 6
analitycznie x(i) = 6.0
i = 5
algorytm x(i) = 9
analitycznie x(i) = 9.0
i = 6
algorytm x(i) = 12
analitycznie x(i) = 12.0
i = 7
algorytm x(i) = 16
analitycznie x(i) = 16.0
i = 8
algorytm x(i) = 20
analitycznie x(i) = 20.0
i = 9
algorytm x(i) = 25
analitycznie x(i) = 25.0
```

Podpunkt 3

Rekurencyjny zapis:

$$\forall n \in \{1, 2, \dots\} : x(n) = x(n-1) + x(n-2)$$
$$x(-1) = x(0) = 0$$

wzór analityczny:

$$x(n) = F(n) = \frac{\phi^n - \frac{(-1)^n}{\phi^n}}{\sqrt{5}} = \frac{(\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n}{\sqrt{5}}$$

In [28]:
```
from math import sqrt

def seq3(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return seq3(n-1) + seq3(n-2)

def seq3_fibbonachi(n):
    return ((((1+sqrt(5))/2)**n - ((1-sqrt(5))/2)**n)/sqrt(5)

for i in range(10):
    print('i =', i)
    print('algorytm x(i) =', seq3(i))
    print('analitycznie x(i) =', seq3_fibbonachi(i))
```

```
i = 0
algorytm x(i) = 0
analitycznie x(i) = 0.0
i = 1
algorytm x(i) = 1
analitycznie x(i) = 1.0
i = 2
algorytm x(i) = 1
analitycznie x(i) = 1.0
i = 3
algorytm x(i) = 2
analitycznie x(i) = 2.0
i = 4
algorytm x(i) = 3
analitycznie x(i) = 3.0000000000000004
i = 5
algorytm x(i) = 5
analitycznie x(i) = 5.000000000000001
i = 6
algorytm x(i) = 8
analitycznie x(i) = 8.000000000000002
i = 7
algorytm x(i) = 13
analitycznie x(i) = 13.000000000000002
i = 8
algorytm x(i) = 21
analitycznie x(i) = 21.000000000000004
i = 9
algorytm x(i) = 34
analitycznie x(i) = 34.00000000000001
```