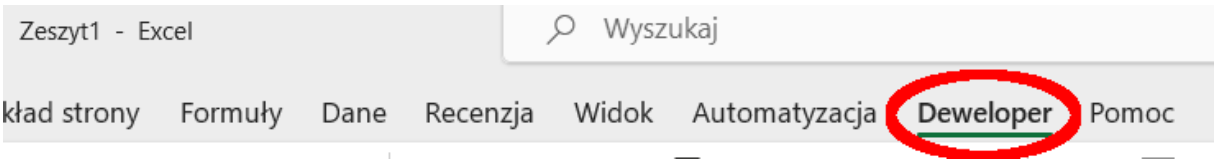
Metody Matematyczne w Procesach Przemysłowych (kierunek: technologia chemiczna)
Wprowadzenie do VBA

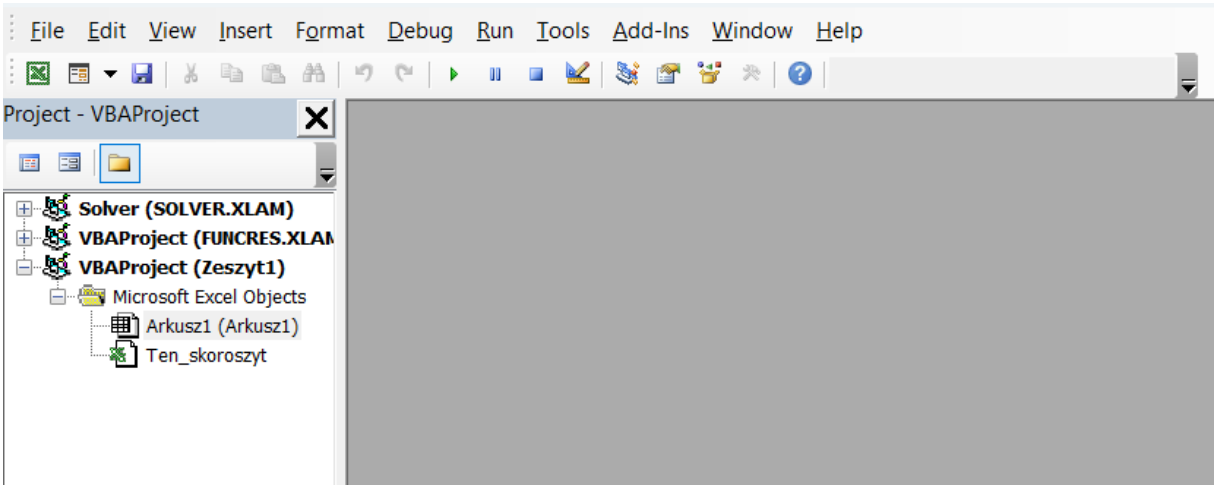
Aby wygodnie pracować z VBA w środowisku arkusza kalkulacyjnego Excel należy włączyć zakładkę **Deweloper** w menu głównym (domyślnie nie jest aktywna):



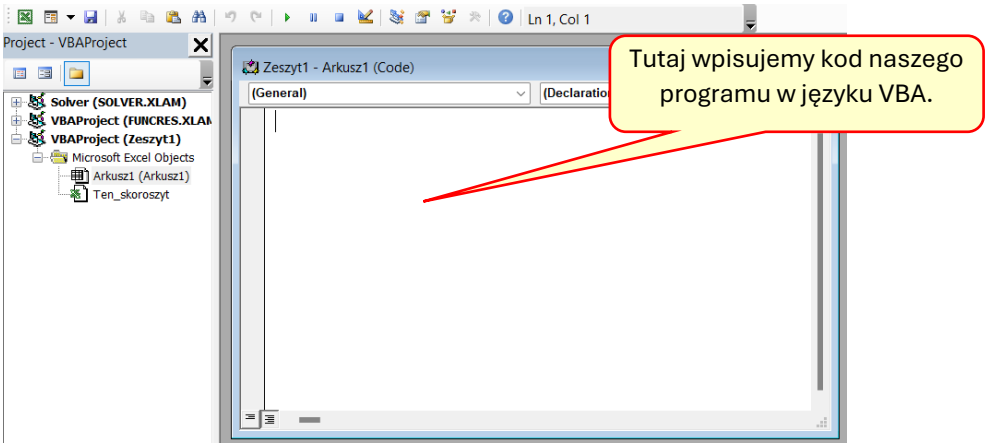
W zależności od wersji pakietu Office włączanie tej zakładki może odbywać się nieco odmiennie. Ale zawsze wymaga to po prostu zaznaczenia odpowiedniej opcji w ustawieniach programu Excel. Na przykład w wersji Office 365 odbywa się to wg ścieżki:

Plik → Opcje → Dostosowywanie Wstążki → Dostosuj Wstążkę (Karty główne) → Deweloper

Środowisko VBA dostępne też jest bez aktywnej karty **Deweloper** – wystarczy użyć skrótu klawiszowego Alt+F11, aby uruchomić środowisko. Ekran, który pojawi się po uruchomieniu VBA w Excelu będzi wyglądał mniej więcej następująco:



Po lewej stronie jest panel projektu VBA („*Project – VBAProject*”), który ma formę drzewa. Domyślnie powinien być rozwinięty folder „*Microsoft Excel Objects*”, w którym aktywny będzie węzeł „*Arkusz1* (*Arkusz1*)”. Należy teraz podwójnie kliknąć na węzeł „*Arkusz1* (*Arkusz1*)”, co spowoduje pojawienie się po prawej stronie okienka edycji kodu VBA (tzw. moduł arkusza):



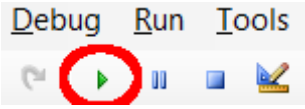
Kod funkcji i procedur wpisujemy w tym oknie. Procedury te będę bezpośrednio dostępne w arkuszu, z którym dany moduł jest związany.

Metody Matematyczne w Procesach Przemysłowych (kierunek: technologia chemiczna)
Wprowadzenie do VBA

Przykład 1. Procedura, która wyświetla komunikat: „Witaj, świecie!”. W okienku edycji kodu VBA wpisujemy następujący kod:

```
Sub hello()
    MsgBox "Witaj, świecie!"
End Sub
```

Aby uruchomić tę procedurę możemy kliknąć przycisk Run Sub lub użyć klawisza F5:



Przykład 2. Rozwiążemy równanie liniowe $ax+b=c$, gdzie $a,b,c\in\mathbb{R}$ są dane. Należy w programie zadeklarować tzw. *zmienne*, które będą służyły do przechowywanie wartości typu liczba rzeczywista. Deklarowanie zmiennych w VBA odbywa się przy pomocy słowa kluczowego **`Dim`**. Na przykład zdanie w VBA:

```
Dim a As Double
```

oznacza, że symbol „a” jest zmienną typu **`Double`** (liczba zmiennopozycyjna o podwójnej precyzji).

Ponadto w naszym programie będziemy używać instrukcji warunkowych, gdyż sposób obliczania rozwiązania zależy od tego czy $a=0$, czy $a\neq 0$. W VBA instrukcja warunkowa **`If`** ma kilka wariantów. Na przykład

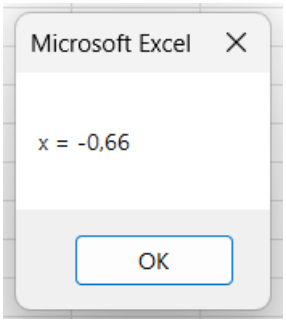
`If` *warunek* **`Then`** *instrukcja*

`If` *warunek* **`Then`**
  Instrukcje
`End If`
`If` warunek **`Then`**
  Instrukcje1
`Else`
  Instrukcje2
`End If`

Program może zatem wyglądać następująco

```
Sub równanie_liniowe_wer1()
Dim a As Double
Dim b As Double
Dim c As Double
Dim x As Double

a = 5
b = 8
c = 4.7

If a <> 0 Then
   x = (c - b) / a
   MsgBox "x = " & x
Else
   MsgBox "Równanie sprzeczne lub ma nieskończenie wiele rozwiązań"
End If
End Sub
```

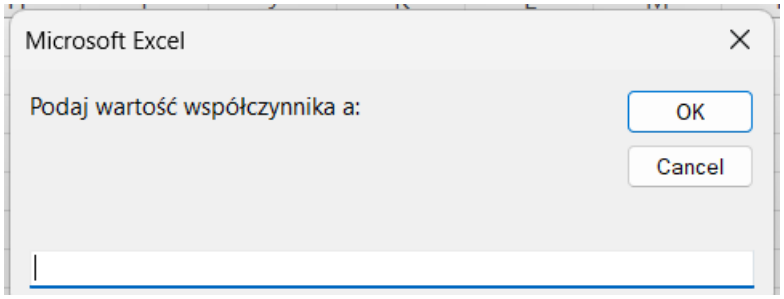
Zwróćmy uwagę na użycie funkcji `MsgBox`. W tym przypadku łańcuch znaków do wypisania w oknie komunikatu ma postać: `"x = " & x`. Zastosowano w nim operator sklejania łańcuchów znaków (&). Ponadto ten fragment, który jest w cudzysłowie jest traktowany dosłownie. Zatem komunikat będzie zawierał na[pis `"x ="`, a dalej będzie aktualna wartość liczbowa zmiennej `x`.



Aktualna wersja jest o tyle niewygodna, że wartości współczynników a, b, c należy podawać bezpośrednio w kodzie programu. Możemy zmienić to używając funkcji VBA o nazwie **InputBox**. Pozwala ona w trybie dialogu z użytkownikiem wprowadzić wartość, na przykład instrukcja

```
a = InputBox("Podaj wartość współczynnika a:")
```

wygeneruje okienko dialogowe



Mamy zatem następującą wersję programu

```
Sub równanie_liniowe_wer2()
Dim a As Double
Dim b As Double
Dim c As Double
Dim x As Double

a = InputBox("Podaj współczynnik a:")
b = InputBox("Podaj współczynnik b:")
c = InputBox("Podaj współczynnik c:")

If a <> 0 Then
   x = (c - b) / a
   MsgBox "x = " & x
Else
   MsgBox "Równanie sprzeczne lub ma nieskończenie wiele rozwiązań"
End If
End Sub
```

W praktyce jednak dla naszych zastosowań, dane najlepiej jest przechowywać w komórkach arkusza i stamtąd je pobierać do programu, który wykonuje obliczenia. Służ do tego funkcja **Cells**. Jej argumenty to numer wiersza i kolumny, np. `Cells(2,5)`. Jeżeli chcemy odczytać wartość z współczynnika „a" z komórki A2, to użyjemy konstrukcji

```
a = Cells(1, 2)
```

Mamy teraz kolejną wersję programu. Proszę też zwrócić uwagę na dane w arkuszu.

	A	B
1	a	3
2	b	8
3	c	-7,5
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		

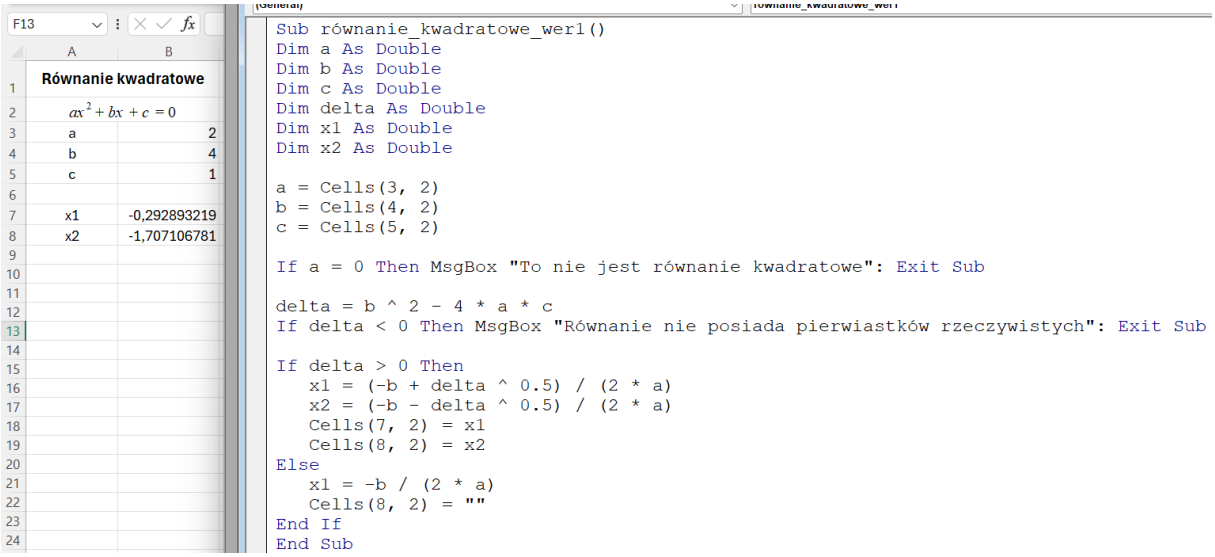
```
Sub równanie_liniowe_wer3()
Dim a As Double
Dim b As Double
Dim c As Double
Dim x As Double

a = Cells(1, 2)
b = Cells(2, 2)
c = Cells(3, 2)

If a <> 0 Then
   x = (c - b) / a
   MsgBox "x = " & x
Else
   MsgBox "Równanie sprzeczne lub ma nieskończenie wiele rozwiązań"
End If
End Sub
```

Nasz program jeszcze wymaga uzupełnienia. Mianowicie w przypadku, gdy $a = 0$, mamy dwie możliwości: $b = c$ (wtedy równanie jest tożsamościowe) albo $b \neq c$ (wtedy równanie jest sprzeczne). Pełna wersja programu jest poniżej

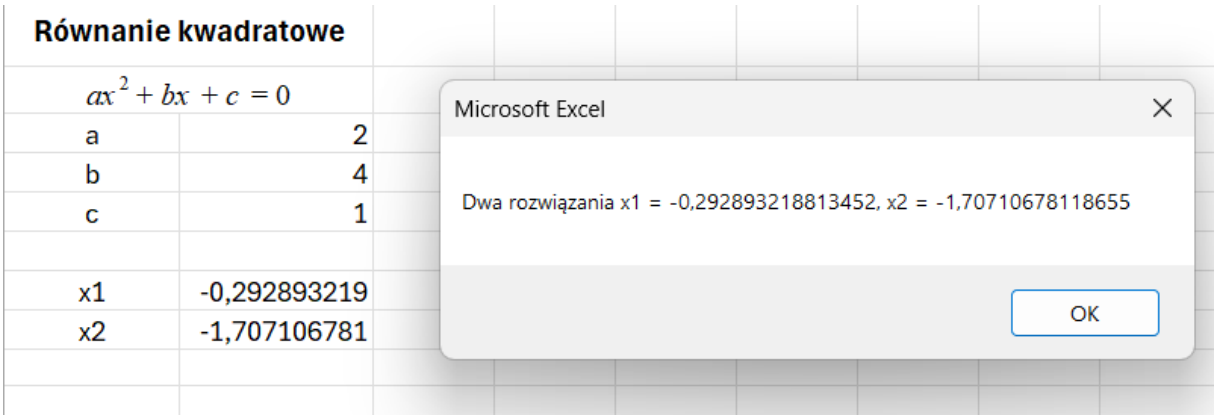
```
Sub równanie_liniowe_wer4()
Dim a As Double
Dim b As Double
Dim c As Double
Dim x As Double

a = Cells(1, 2)
b = Cells(2, 2)
c = Cells(3, 2)

If a <> 0 Then
   x = (c - b) / a
   MsgBox "x = " & x
ElseIf b = c Then
   MsgBox "Każda liczba jest rozwiązaniem."
Else
   MsgBox "Równanie sprzeczne."
End If
End Sub
```

Przykład 3. Program będzie rozwiązywał równanie kwadratowe $ax^2 = bx + c = 0, \quad a \neq 0,$ w dziedzinie liczb rzeczywistych. Obsłuży on wszystkie szczególne przypadki.

Podstawowa wersja programy jest przedstawiona poniżej.



Można rozbudować program tak, aby pojawiały się stosowne komunikaty. Na przykład

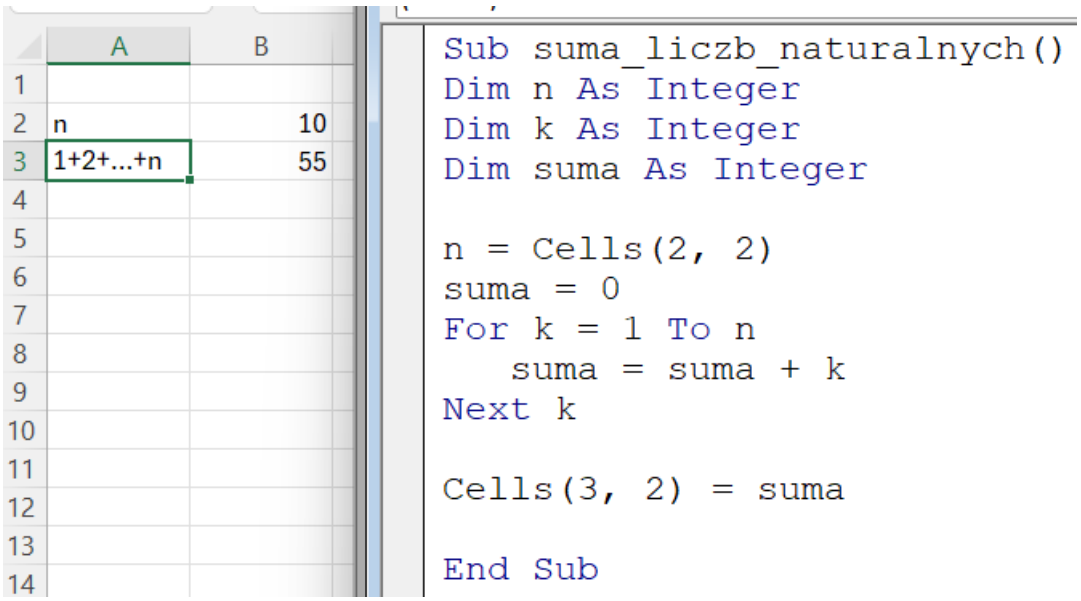


W tym celu dodajemy instrukcje w odpowiednich gałęziach instrukcji **`If-Then-Else`**:

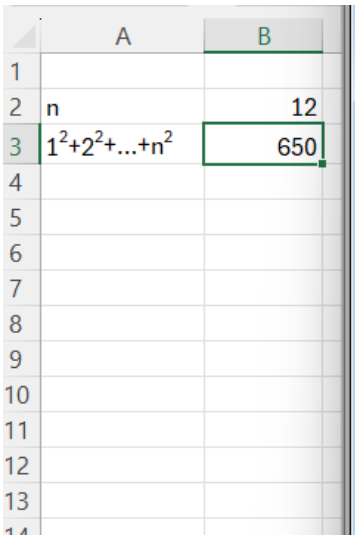
```
If delta > 0 Then
   x1 = (-b + delta ^ 0.5) / (2 * a)
   x2 = (-b - delta ^ 0.5) / (2 * a)
   MsgBox "Dwa rozwiązania x1 = " & x1 & ", x2 = " & x2
   Cells(7, 2) = x1
   Cells(8, 2) = x2
Else
   x1 = -b / (2 * a)
   MsgBox "Jedno rozwiązanie x1 = x2 = " & x1
   Cells(8, 2) = ""
End If
```

Przykład 4. Iteracje służą do powtarzania jakiejś instrukcji lub zestawu instrukcji. W VBA jest kilka wariantów instrukcji iteracyjnych. Nam wystarczą dwie: „**`For`**” oraz „**`While`**”

Sumujemy liczby naturalne od 1 do *n*:



Sumujemy kwadraty kolejnych liczb naturalnych:



	A	B
1		
2	n	12
3	$1^2+2^2+...+n^2$	650
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		

```
Sub suma_kwadratów_liczb_naturalnych()
Dim n As Integer
Dim k As Integer
Dim suma As Integer

n = Cells(2, 2)
suma = 0
For k = 1 To n
   suma = suma + k ^ 2
Next k

Cells(3, 2) = suma

End Sub
```

Przykład 5. Obliczanie pierwiastka kwadratowego metodą Herona.

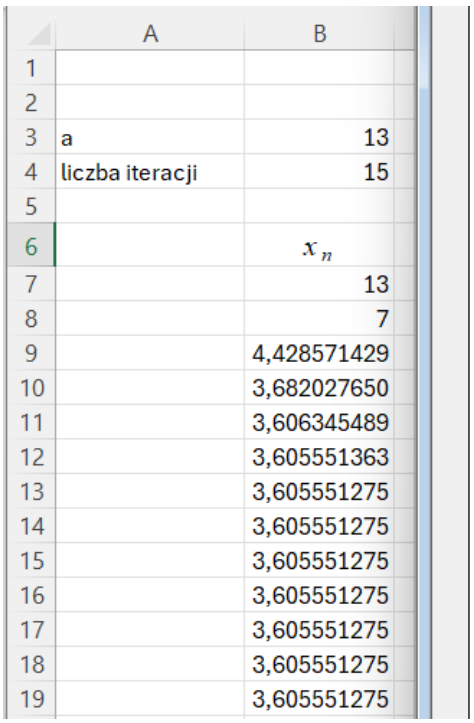
Dana jest liczba $a \geq 0$. Aby obliczyć przybliżenie pierwiastka $\sqrt{a}$ korzystamy z ciągu

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n} \right) \text{ dla } n = 0, 1, 2, 3, \ldots \tag{1}$$

Należy też podać wartość początkową ciągu, x_0. W tym przypadku okazuje się, że ciąg (1) zawsze będzie zbieżny do $\sqrt{a}$, pod warunkiem, że $x_0 > 0$. W praktyce stosuje się wybór

$$x_0 = \begin{cases} a, & \text{gdy } 0 \leq a \leq 1, \\ 1, & \text{gdy } a > 1. \end{cases} \tag{2}$$

Implementacja ciągu (1) z wyborem wyrazu początkowego (2) jest poniżej. Wykonywane jest 15 iteracji, a kolejne wyrazy ciągu są wpisywane do komórek arkusza.



	A	B
1		
2		
3	a	13
4	liczba iteracji	15
5		
6		x_n
7		13
8		7
9		4,428571429
10		3,682027650
11		3,606345489
12		3,605551363
13		3,605551275
14		3,605551275
15		3,605551275
16		3,605551275
17		3,605551275
18		3,605551275
19		3,605551275

```
Sub ciąg_Herona()
Dim a As Double
Dim x As Double
Dim n As Integer
Dim liczba_iter As Integer

a = Cells(3, 2)
liczba_iter = Cells(4, 2)

If a <= 1 Then x = 1 Else x = a

Cells(7, 2) = x
For n = 1 To liczba_iter
   x = 0.5 * (x + a / x)
   Cells(n + 7, 2) = x
Next n
End Sub
```

Przykład 6. W tym przykładzie poznamy działanie operatora **mod**, który zwraca resztę z dzielenia dwóch liczb całkowitych. Na przykład 13 **mod** 5 = 3, 21 **mod** 4 = 1, 8 **mod** 2 = 0. Widać, że aby sprawdzić w VBA czy liczba całkowita $b \neq 0$ dzieli liczbę całkowitą a wystarczy sprawdzić warunek a **mod** b = 0:

```
If a mod b = 0 Then
    ...
End If
```

Ciąg Collatza jest to ciąg liczb naturalnych zdefiniowany następująco:

$$c_0 \in \mathbb{N} \text{ zadajemy},$$
$$c_{n+1} \begin{cases} c_n / 2, & \text{jeżeli } c_n \text{ jest parzysta}, \\ 3c_n + 1, & \text{jeżeli } c_n \text{ jest nieparzysta}, \end{cases}$$

przy czym przerywamy tworzenie nowego wyrazu ciągu, jeśli pojawi się wyraz równy 1. Na przykład dla $c_0 = 6$ mamy ciąg: 6, 3, 10, 5, 16, 8, 4, 2, **1**.

Ponieważ nie wiadomo kiedy pojawi się jedynka dla zadanej wartości wyrazu początkowego, należy użyć pętli warunkowej, na przykład „**While**”. Do sprawdzenia czy liczba jest parzysta użyjemy instrukcji

```
If c mod 2 = 0 Then
```

Prosty program obliczający wyrazy ciągu Collatza i wpisujący je do komórek arkusza:

	A	B
1		
2		
3	Ciąg Collatza	
4	c0 (zadajemy)	6
5		3
6		10
7		5
8		16
9		8
10		4
11		2
12		1
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		

```
Sub Collatz()
Dim c As Integer
Dim k As Integer

k = 4
c = Cells(4, 2)

If c <= 0 Then MsgBox "Wpisz liczbe naturalną, c > 0": Exit Sub

While c <> 1
   If c Mod 2 = 0 Then
      c = c / 2
   Else
      c = 3 * c + 1
   End If

   k = k + 1
   Cells(k, 4) = c
Wend
End Sub
```

Przykład 7. Rozwiązywanie równania $(3-x)e^x = 3$ metodą bisekcji.

Równanie przepisujemy do postaci $f(x) = 0$: $(3-x)e^x - 3 = 0$. Można rozwiązywać metodą Newtona, ale okazuje się, że w tym przypadku trzeba wybrać punkt startu bardzo blisko nieznanego rozwiązania. Zastosowanie metody bisekcji pomija ten problem. Wystarczy tylko wystartować od przedziału na końcach którego funkcja $f(x)$ ma różne znaki. Na przykład przedział [1; 3] jest dobry, gdyż

$$f(1) = 2e^1 - 3 > 2 \cdot 2 - 3 = 1, \quad f(3) = -3.$$

```
Function f(x As Double) As Double
f = (3 - x) * Exp(x) - 3
End Function

Sub bisekcja()
Dim a As Double
Dim b As Double
Dim x As Double
Dim eps As Double

a = 1
b = 3
eps = 0.000001

While (b - a > eps)
   x = (a + b) / 2
   If f(x) = 0 Then MsgBox x: Exit Sub

   If f(a) * f(x) < 0 Then
      b = x
   Else
      a = x
   End If
Wend

MsgBox "x = " & x & ", f(x) = " & f(x)
End Sub
```

Przykład 8. Rozwiązywanie równania nieliniowego $f(x)=0$ metodą Newtona. Metoda ta polega na tworzeniu ciągu $(x_n)_{n=0}^{\infty}$ w oparciu o wzór

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, 2, \ldots$$

Jeżeli wyraz początkowy x_0 jest dostatecznie blisko rozwiązania, to powyższy ciąg jest zbieżny do tego miejsca zerowego.

Zastosujemy tę metodę do obliczenia miejsca zerowego równania

$$x^3 + K_a x^2 - (K_w + c_0 K_a)x - K_a K_w = 0,$$

gdzie K_a, K_w, c_0 to dodatnie dane parametry.

Rozwiązanie opiera się na zdefiniowaniu dwóch funkcji w języku VBA: jedna definiuje równanie, a druga jego pochodną:

```
Function f(x As Double) As Double
f = x ^ 3 + Ka * x ^ 2 - (Kw + Ka * c0) * x - Ka * Kw
End Function

Function Df(x As Double) As Double
Df = 3 * x ^ 2 + 2 * Ka * x - (Kw + Ka * c0)
End Function
```

W definicji tych funkcji odwołujemy się do stałych K_a, K_w, c_0, które powinny być gdzieś zdefiniowane w programie. Najlepiej będzie w tym przypadku, aby były to tzw. stałe, których wartości będę zainicjalizowane na początku programu. Całość programu jest poniżej.

```
Const Ka = 0.000012
Const Kw = 0.00000000000001
Const c0 = 0.000005

Function f(x As Double) As Double
f = x ^ 3 + Ka * x ^ 2 - (Kw + Ka * c0) * x - Ka * Kw
End Function

Function Df(x As Double) As Double
Df = 3 * x ^ 2 + 2 * Ka * x - (Kw + Ka * c0)
End Function

Sub metoda_Newtona()
Dim x As Double
Dim n As Integer

x = 0.5 * c0

For n = 1 To 20
   x = x - f(x) / Df(x)
Next n

MsgBox "x = " & x & ", f(x) = " & f(x)
End Sub
```

Wyniki, który uzyskujemy po uruchomieniu tego programu jest następujący: