

Efektivní procházení jednotlivých buněk v cyklu je nejčastější úlohou programování ve VBA. V tomto článku se podíváme na to, jak projít buňky dané svým umístěním, nikoliv obsahem.

Mějme úkol projít oblast buněk ve sloupci A dle obrázku.

	A	B	C	D	E	F	G	H	I
1									
2	a	<i>Co nejefektivnějším způsobem vyberte v cyklu vyznačené buňky.</i>							
3									
4									
5	b								
6									
7									
8	c								
9									
10									
11	d								
12									
13									
14	e								
15									
16									
17	f								
18									
19									
20									
21									

Efektivní procházení jednotlivých buněk v cyklu - úloha

Zapomeňme teď na to, že buňky jsou specifické svým obsahem a mohli bychom využít dialogů Najít a Přejít na / Jinak... (kolekce SpecialCells). Hodnoty buněk nyní slouží pouze pro vizuální orientaci a my si máme všimnout jejich pořadí. Je vidět, že až na umístění první buňky je mezi nimi odstup dvou buněk (a budeme předpokládat, že by toto chování platilo i pro další buňky ve sloupci). Jak bychom takovou oblast prošli? Už asi tušíte, že cílem není v rámci cyklu $x = 1$ až 17 projet všechny buňky a kdovíjakým způsobem je testovat a zpracovávat. My potřebujeme zacílit přímo na 6 konkrétních buněk ze 17 a tedy i cyklus vykonat pouze šestkrát a nikoli sedmnáctkrát (což je téměř třikrát více). Vezměme si tužku a papír nebo zapisujme přímo do tabulky, na jaké buňky (řádky) se musíme dostat v rámci jednotlivých cyklů (kroků):

- 1 → 2
- 2 → 5
- 3 → 8
- 4 → 11
- 5 → 14
- 6 → 17

V podstatě máme dvojici x a $y=f(x)$, čili nezávisle proměnnou x (představující krok, čítač cyklu) a závisle proměnnou y . Jakého typu je vazba mezi nimi? Věřte tomu, že v těchto typech úloh se jedná o lineární závislost (přímku) a uplatní se tedy rovnice $y = ax+b$, jak ji známe ze základní školy. Know-how úlohy tak spočívá ve zjištění koeficientů a a b . Nejjednodušším způsobem je označit data v tabulce a vložit XY bodový graf. Následně vybrat body řady a pod pravým tlačítkem zvolit Přidat spojnicí trendu. Její typ bude lineární. Nejdůležitější je nechat si zobrazit rovnici v grafu a také hodnotu spolehlivosti.

The screenshot displays an Excel spreadsheet with the following data and formulas:

x	y = f(x) = ?
1	2
2	5
3	8
4	11
5	14
6	17

	Směrnice přímky (a)	Průsečík s osou y (b)
Lineární regrese	3,0000	-1,0000
$y = ax + b$	3,0000	-1,0000

Formulas in the spreadsheet:

- C12:D12: {=LINREGRESE(D3:D8;C3:C8;PRAVDA)}
- C13: =SLOPE(D3:D8;C3:C8)
- D13: =INTERCEPT(D3:D8;C3:C8)
- C14: =INDEX(LINREGRESE(D3:D8;C3:C8);1)
- D14: =INDEX(LINREGRESE(D3:D8;C3:C8);2)
- D21: =STEYX(D3:D8;C3:C8)

The chart shows a linear trendline with the equation $y = 3x - 1$ and $R^2 = 1$.

The 'Format trendline' task pane shows the following options:

- Možnosti spojnice trendu:
 - Exponenciální
 - Lineární
 - Logaritmická
 - Polynomičná (Stupeň: 2)
 - Mocnná
 - Klouzavý průměr (Perioda: 2)
- Název spojnice trendu:
 - Automaticky (Lineární (řada1))
 - Vlastní
- Odhad:
 - Dopředu: 0,0 per.
 - Dál: 0,0 per.
 - Hodnota Y: 0,0
- Zobrazit rovnici v grafu
- Zobrazit hodnotu spolehlivosti R

Excel - lineární regrese

Pokud je hodnota spolehlivosti rovna 1 (všechny body evidentně leží na přímce), máme vyhráno. Zjištěnou závislost si můžeme ověřit přímo v tabulce hodnot.

Obrázek ukazuje ještě cestu bez grafu, s využitím funkcí listu. Hodnotu spolehlivosti na listu nahrazuje standardní chyba.

A co s výsledkem? Není proč otálet, zapracujeme jej do kódu VBA. Pro klasický čítač a procházení jednotlivých buněk se samozřejmě hodí kolekce Cells se syntaxí Cells(řádek, sloupec).

1	Sub ProchazeniBunek()
2	
3	Dim x As Integer
4	
5	For x = 1 To 6
6	
7	Cells(3 * x - 1, 1). Select
8	
9	Next x
10	
11	End Sub

Úloha byla postavena tak, že by v ní zkušenější oko vidělo vztah od boku. Nicméně i když se jedná o přímkovou závislost, ne vždy jsou koeficienty a a b zřejmé na první pohled. Pokud v jiných typech úloh budete chtít využití lineární regrese, můžete se stát, že koeficienty nebudou celočíselné (zde to nehrozí). V tom případě vám budiž nápomocná tabulka pro typické zlomky – viz obrázek.

Procházení jednotlivých buněk s pomocí metody Offset

Stejný postup použijete v případě, kdy z nějaké nějaké startovní buňky budete „skákat“ na cílové buňky. Tehdy se ve VBA uplatní metoda Offset, která (spolu s metodou Resize) odpovídá činnosti funkce POSUN na listu. Nezapomeňte, že metoda Offset představuje slovní „posun o ...“, kdežto Resize řeší změnu „velikosti na ...“.

Můžete si vyzkoušet, že pro startovní buňku A1 a metodu Offset je závislost $y = 3x - 2$.

```
1 Sub ProchazeniBunekOffset()  
2  
3 Dim x As Integer  
4  
5 For x = 1 To 6  
6  
7 Cells(1, 1).Offset(3 * x - 2).Select  
8  
9 Next x  
10  
11 End Sub
```

Aby byl kód ještě efektivnější, bylo by vhodné startovní buňku vyjmout z cyklu, pojmenovat (přiřadit do objektové proměnné), a aplikovat konstrukci With..End With.

A co když posloupnost procházených buněk žádnou logiku nemá?

	A	B
1		
2		
3	a	
4		
5		
6	b	
7	c	
8		
9		
10		
11	d	
12		
13		
14		

Efektivní procházení
buněk v cyklu - úloha

V takovém případě asi nezbývá, než si pořadí položek definovat v poli.

1	Sub ProchazeniBunekVycet()
2	
3	Dim arrPole()
4	
5	arrPole = Array(3, 6, 7, 11)
6	
7	For Each Index In arrPole
8	
9	Cells(Index, 1). Select
10	
11	Next Index
12	
13	End Sub

Příloha

[efektivni_prochazeni_bunek.zip](#)