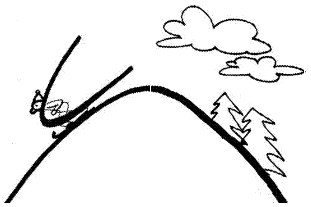
INFORMATIKA

Animované grafy v Excelu

ŠTĚPÁN HUBÁLOVSKÝ,

Přírodovědecká fakulta UHK, Hradec Králové



Článek ukazuje možnosti vytvoření jednoduchých animovaných grafů v aplikaci MS Excel. Tyto animované grafy lze využívat jako pomocný prostředek pro znázornění průběhu funkcí v matematice. V článku je popsán princip iteračního přepočtu buňky v Excelu, na jehož základě se animované grafy vytvářejí.

Animace

Animace obecně znamená způsob, jak zdánlivě „rozhýbat" statický obrázek. Principem animace je zaznamenání sekvence snímků, které se nepatrně liší. Při rychlém zobrazování těchto snímků za sebou vzniká díky setrvačnosti lidského oka dojem pohybu. Způsob, jak animovat graf, ukážeme na příkladu animace průběhu kvadratické funkce $f(x) = a(x+b)^2+c$. Animovat graf znamená, že změnou jednoho, popř. více parametrů, dojde ke změně zdrojových dat grafu a graf krok za krokem bude definovaně měnit svůj tvar. Pokud tedy budeme měnit např. parametr c (při konstantních parametrech a a b), bude se tento graf posouvat ve směru osy y – viz např. [1]. Nyní si ukážeme, jak zajistíme změnu parametru c pomocí tzv. iterativního přepočtu hodnoty buňky.

Iterace

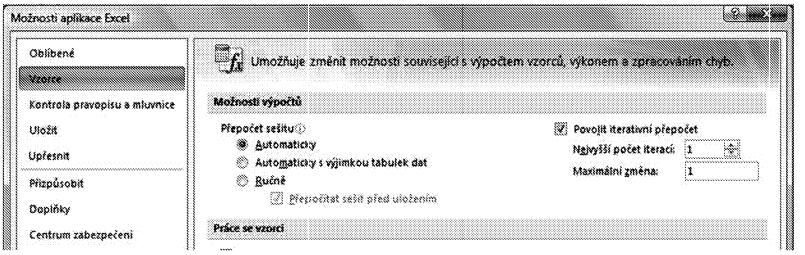
Obecně iterace znamená opakování určité činnosti – slovo iterace pochází z latinského *iterare* – opakovat. V matematice a numerických meto-

dách iterace je proces opakovaného použití funkce, kdy v dalším kroku se funkční hodnota použije jako argument této funkce – pomocí matematické symboliky zapsáno: $x_{n+1} = f(x_n)$ – viz např. [3]. Typickým příkladem využití iterací je numerické řešení nelineárních rovnic obecnou iterační metodou – viz např. [2].

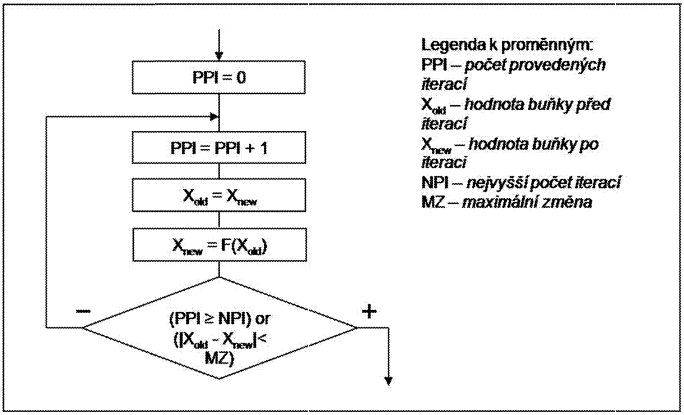
Iterace v programování, podobně jako v matematice, znamená opětovné volání funkce, kdy argumentem funkce je výsledek předešlého volání funkce. V programování se s iteracemi setkáváme poměrně často. I přiřazovací příkaz typu A:= A + 1 lze chápat jako formu iterace. Tento výraz znamená, že nová hodnota proměnné je rovna původní hodnotě A zvýšené o 1 ($A_{N+1} := A_N + 1$).

Iterace v Excelu

V Excelu se ve vzorci dané buňky může vyskytnout cyklický odkaz. Znamená to, že vzorec v buňce se odkazuje na hodnotu této buňky, popř. že vzorec této buňky se odkazuje na hodnotu jiné buňky, která má ve vzorci opět odkaz na původní buňku. Hodnotu takové buňky lze stanovit pouze iterativním přepočtem. Pokud v nástrojích Excelu není povolen iterativní přepočet (standardně je zakázán), výpočet se neprovede a bude oznámena chyba. Ve verzi Excelu 2007 se iterativní přepočet povolí v dialogovém okně *Možnosti aplikace Excel* (tlačítko na otevření dialogového okna se objeví po kliknutí na Tlačítko Office) v záložce *Vzorce*, což je patrno z obr. 1. Zde je třeba zaškrtnout volbu *Povolit iterativní přepočet*. Položka *Nejvyšší počet iterací* uvádí maximální počet iterací, opakování, které Excel provede při iterativním přepočtu. Položka *Maximální změna* stanovuje přesnost výpočtu – porovnává se zde hodnota buňky před iterací a hodnota



Obr. 1 Nastavení hodnot pro iterativní přepočet



Obr. 2 Část vývojového diagramu popisující iterativní přepočet

buňky po iteraci. Pokud absolutní rozdíl obou hodnot je menší než stanovená přesnost výpočtu, bude iterace zastavena. Standardně jsou v Excelu nastaveny hodnoty *Nejvyšší počet iterací* na 100 a *Maximální změna* na 0,001. Iterativní výpočet a jeho ukončení lze pro snazší pochopení vyjádřit formou algoritmu zapsaného vývojovým diagramem na obr. 2. Iterativní přepočet lze ručně spustit stiskem klávesy F9 a to i přesto, že v dialogovém okně *Možnosti aplikace Excel* je Nastavena volba *Automaticky.*

Nastavení iteračního přepočtu pro animaci grafu.

Vraťme se nyní k využití iteračního přepočtu pro animování grafu. Konkrétně budeme měnit např. parametr c kvadratické funkce $f(x) = a(x+b)^2 + c$. Hodnotu tohoto parametru uložíme do buňky B4. Do této buňky dále napíšeme cyklický vzorec „=B4+1“, který vyjadřuje, že hodnota buňky B4 se po provedení jednoho iteračního přepočtu navýší o 1. Ručním provedením přepočtu stiskem klávesy F9 se provede tolik přepočtů, kolik udává hodnota stanovená v okně *Maximální počet iterací.* Chceme-li provést jeden, vzorcem v buňce B4 definovaný přepočet, je nutné tuto hodnotu nastavit na 1, jak je zřejmé z obrázku 1. Nyní opakovaným stiskem, popř. držením klávesy F9 se hodnota buňky B4 bude opakovaně zvyšovat o hodnotu 1. Pokud by v buňce B4 byl uveden pouze vzorec „=B4+1“, hodnota B4 by se po každém přepočtu stále zvyšovala. Dokonce i při uložení souboru, jeho uzavření a následném otevření se provede vždy jeden přepočet.

Je proto vhodné stanovit minimální mez – je udaná hodnotou v buňce C2, a maximální mez – je udaná hodnotou v buňce D2, v rámci kterých se budou přepočtené hodnoty B4 pohybovat. To lze zajistit tak, že vzorec v buňce B4 bude doplněn podmínkou: „=KDYŽ(B4>=D2;C2;B4+E2)“ – obrázek 5. Tato podmínka porovnává hodnotu v buňce B4 s hodnotou v buňce D2. Pokud je podmínka (B4>=D2) splněna, hodnota v buňce B4 se změní na hodnotu v buňce C2. V opačném případě se provede iterační přepočet a hodnota buňky B4 se zvýší o hodnotu uloženou v buňce E2 – velikost iteračního kroku. Opakovaným stiskem, popř. držením klávesy F9 se bude provádět výše definovaný přepočet parametru c kvadratické funkce v mezích od minimální (D2) do maximální hodnoty (C2) se zadaným iteračním krokem (E2).

Grafy v Excelu

Tabulkový editor MS Excel má poměrně dobře implementované nástroje pro tvorbu grafů. Umožňuje na základě požadavků uživatele vytvářet různé typy grafů, jejichž úplný seznam lze najít v nápovědě Excelu, popř. v nabídce *Vložení grafu*. V tomto článku se budeme zabývat pouze grafy XY bodovými, které umožňují v rovině zobrazit množinu bodů daných souřadnicemi x a y. Při „dostatečném“ množství „hustě“ zvolených bodů lze pomocí tohoto typu grafu zobrazovat průběhy zvolených funkcí.

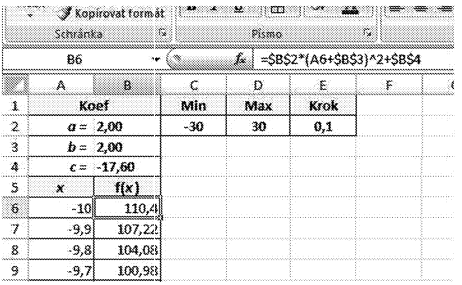
Příprava hodnot pro XY bodový graf

Jak již bylo uvedeno výše, animaci grafu si ukážeme na grafu zvolené kvadratické funkce. Tento graf zobrazíme v intervalu pro $x \in \langle -10; 10 \rangle$ Pomocí iterativního přepočtu se bude měnit koeficient c, jehož hodnota je uložena v buňce B4, koeficienty a, b budou neměnné a jsou uloženy v buňkách B2 a B3. Do jednoho ze sloupců (např. A) vepíšeme pod sebe souřadnice x všech zobrazovaných bodů v daném intervalu. S ohledem na přesnost zobrazení je nutné zvolit tyto souřadnice poměrně „hustě“ vedle sebe, např. po 0,1. S výhodou zde využijeme možnosti automatického kopírování řady čísel tažením za úchyt buňky. Počet takto vytvořených bodů je 201. Ve sloupci B provedeme výpočet hodnot funkce $f(x)$. Vzorec uložený v buňce B6 je pak zřejmý z obr. 3. Odkazy na koeficienty a, b, c buňkách B2, B3 a B4 jsou absolutní, odkazy na hodnoty argumentu funkce ve sloupci A jsou relativní. Vytvořený vzorec tažením za úchyt překopírujeme z buňky B6 do buněk B7 až B206.

Vytvoření a animace grafu

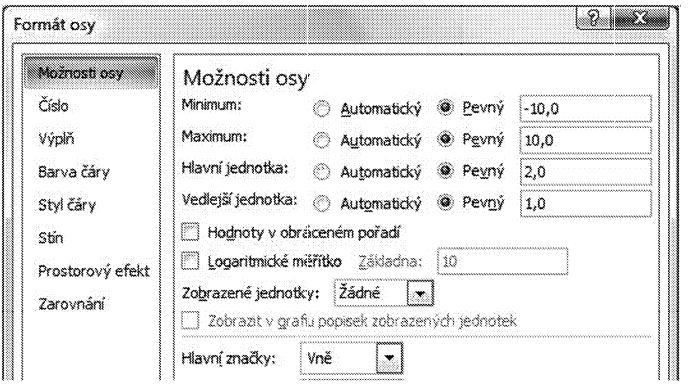
Vybereme data, ze kterých budeme vytvářet graf – v našem případě

oblast A6:B206, a vložíme bodový graf s vyhlazenými spojnicemi (tento typ grafu je určen ke kreslení funkcí). Na zobrazeném grafu je vhodné nastavit maximální a minimální meze obou os pevně, v opačném případě animace automaticky mění tyto hodnoty a výsledek není tak názorný. Ve verzi Excel 2007 se nastavení provádí v dialogovém okně *Formát osy*, zvlášť pro osu x, zvlášť pro osy y – obr. 4.



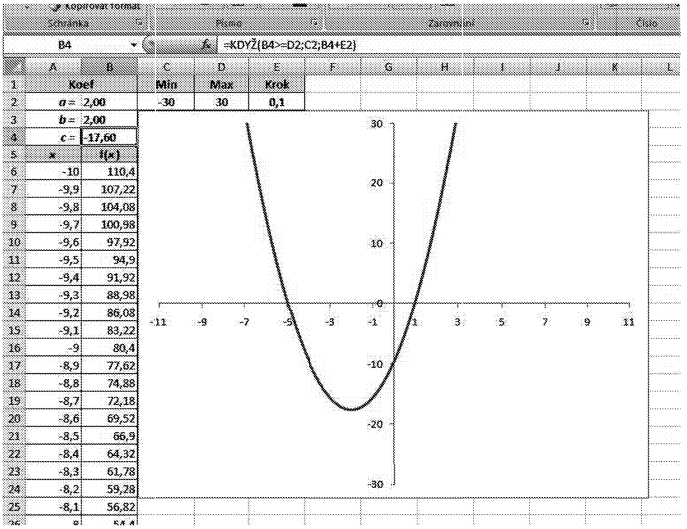
	A	B	C	D	E	F
1	Koef		Min	Max	Krok	
2	*a* =	2,00	-30	30	0,1	
3	*b* =	2,00				
4	*c* =	-17,60				
5	*x*	f(x)				
6	-10	110,4				
7	-9,9	107,22				
8	-9,8	104,08				
9	-9,7	100,98				

Obr. 3 Vzorec v buňce B6 pro graf kvadratické funkce



Obr. 4 Nastavení vlastností osy grafu

Po každém provedeném iterativním přepočtu koeficientu c dané kvadratické funkce v mezích a s krokem tak, jak bylo uvedeno v předešlém textu, dojde k přepočítání funkčních hodnot v oblasti B6:B206 a zároveň dojde k automatickému posunu statického grafu, graf funkce se posouvá ve směru osy y. Vhodnou volbou velikosti iteračního kroku (hodnota buňky E2) se graf jeví jako animovaný – obr. 5.



Obr. 5 Animovaný graf kvadratické funkce vytvořený iterativním přepočtem buňky B4

Závěr

Článek se zabývá možností využití ICT ve výuce matematiky. Na konkrétním příkladu animace kvadratické funkce je ukázáno využití iterativního přepočtu, implementovaného do tabulkového editoru MS Excel, jako nástroje pro vytvoření jednoduchých animovaných grafů. Podobně lze aplikovat metodu animace pro parametry a a b, popř. současně měnit více parametrů najednou. Metodu animace lze použít samozřejmě i v jiných předmětech i pro složitější funkční závislosti, dokonce i pro grafické řešení fyzikálních modelů.

Literatura

[1] *Odvárko, O.*: Matematika pro gymnázia, Funkce, Prometheus, 1993.

[2] *Jehlička, V.*: Numerické metody, 4. lekce: Řešení nelineárních rovnic. Dostupné na http://lide.uhk.cz/pdf/ucitel/jehlivl1/vyuka/numeraky/numericke_metody.htm

[3] *Redakce MFI*: Odmocnina. MFI, r. 18 (2008/09), č. 1, s. 52–53.

(Autorkou úvodní ilustrace je *Mgr. Jaroslava Čermáková.*)

Součet dělitelů a počítač

STANISLAV TRÁVNÍČEK

Přírodovědecká fakulta UP, Olomouc

Přirozená čísla, ta přece zná každý, každý dovede odříkávat $1, 2, 3, 4, \ldots$, co na nich může být zajímavého kromě toho, že vyjadřují počet (mamutů, knih, desetinných míst)? Tak se někdo může ptát při povrchním pohledu. Ale pohleďme trochu hlouběji.

Přirozená čísla postupně vznikala už v prvních fázích komunikace našich prapředků a jen co se osvobodila od počítaných předmětů (tj. vznikly pojmy „jedna“, „dvě“, atd.), zaujala mnoho přemýšlivých a hravých předchůdců pozdějších matematiků. V průběhu historie se jimi zabývali i ti největší matematikové a dodnes jsou tato čísla předmětem hlubokých studií. O jejich vlastnostech jsou napsány rozsáhlé spisy, ale o obsah těchto spisů nám dnes nepůjde. Zkusíme, jaké by to bylo a jaký bychom měli pocit, kdybychom chtěli přijít na něco nového (i když to třeba objektivně nové nebude).

V předloženém článku pojednáme o jednom zajímavém problému, který se týká dělitelů a jejich součtů. Nahlédneme, jak může počítač být nápomocen při získávání úvodních poznatků o nějakém problému, na jejichž základě může matematik vytvářet netriviální hypotézy.

Součet dělitelů

V rozvinutých kulturách již dávno žili lidé, kteří se zajímali o přirozená čísla, a kteří si postupně uvědomovali, že některá čísla jsou součinem jiných čísel a jiná ne, takže se začala rozlišovat čísla složená a prvočísla. Při studiu složených čísel vytvářeli množiny všech dělitelů daného čísla n, např. číslo 12 má dělitele 1, 2, 3, 4 a 6 (také 12, ale budeme zde uvažovat jen dělitele menší než n a nebudeme se zabývat případem $n = 1$). A tu někoho nejméně před dvěma a půl tisíci lety napadlo všechny dělitele daného čísla sečíst. Proč, to je otázka, protože kromě lidské zvídavosti pro to snad rozumný důvod ani nebyl. Ale postupně narůstalo poznání o přirozených číslech a matematici se mnoha problémy týkajícími se dělitelnosti a dělitelů i součtu dělitelů zabývají dodnes. Podívejme se na tento problém z programátorského hlediska.

Součet dělitelů daného čísla n označme $s(n)$, tedy např. $s(12) = 1+2+3+4+6 = 16$, $s(10) = 1+2+5 = 8$, $s(6) = 1+2+3 = 6$. Pro výpočet $s(n)$ na počítači vytvoříme (v Pascalu) funkci

```
function SD(M: Longint): Longint;
var
  S, K: Longint;
begin
  K := 2; S := 1;
  while M >= K * K do
  begin
     if M mod K = 0 then
         S := S + K + (M div K);
       if M = K * K then
         S := S - K;
     Inc(K)
   end;
   SD := S
end;
```

Na počátku zadáme velikost součtu S dělitelů hodnotou $S = 1$ (tím započítáváme samozřejmého dělitele 1) a do S postupně akumulujeme další dělitele. V cyklu počínaje $K = 2$ a kde K postupně zvyšujeme o jedničku, dělíme zadané číslo M číslem K a pokud dělení vyjde beze zbytku, přičteme k S nejen dělitele K, ale i dělitele M/K. Proto stačí uvažovat K jen do hodnoty $\sqrt{M}$, tj. cyklus se provádí, pokud $M \geq K^2$. Když vyjde přesně $M = K^2$, např. pro $M = 25$ a $K = 5$, byl by dělitel K započítán dvakrát ($M/K = K$), tak posledního dělitele K jednou odečteme (ponecháváme čtenáři, aby vymyslel i jiný způsob ošetření případu $M = K^2$).

Pro zajímavost poznamenejme, že na internetu v části „Dokonalá čísla“ je v ukázce použit jiný algoritmus. V cyklu se k S přičítá jen právě zjištěný dělitel K a pátrání po dělitelech končí až pro $K = M/2$. Tato odlišnost velmi zpomaluje výpočty; jeho časová složitost je $\mathrm{O}(M/2)$ na rozdíl od výše uvedeného algoritmu se složitostí $\mathrm{O}(\sqrt{M})$. Pro větší M je to velký rozdíl. Např. pro $M = 1\,000\,000$ se počet průběhů cyklem zmenší z 500 000 při $\mathrm{O}(M/2)$ na pouhý 1 000 v našem případě. Uvedenou funkci $SD(M)$ zapracujeme do všech dalších programů v tomto článku.

Jako první si uveďme zcela jednoduchý program na výpočet součtu

dělitelů čísel n v zadaném rozmezí od A do B.

```
program SoucetD;
var
  A, B, N: Longint;

function SD(M: Longint): Longint;
...

begin    {program}
  WriteLn('Soucet delitelu cisel n');
  Write('od A = ');
  ReadLn(A);
  if A = 1 then A := 2;
  Write('do B = ');
  ReadLn(B);
  N := A;
  while N <= B do
  begin
    WriteLn(N,'  ', SD(N));
    if (N mod 20) = 0 then ReadLn;
    Inc(N)
  end;
  ReadLn
end.     {program}
```

Tento program po výstupu každých dvaceti řádků práci zastaví, aby měl uživatel dostatek času k prohlédnutí nových výsledků. Jeho vedlejším použitím může být také zjišťování, zda dané číslo n je prvočíslo (v tomto případě je $s(n) = 1$), i když samozřejmě mnohem výhodnější jsou k tomuto účelu programy, o jakých jsme pojednávali v redakčním článku [1].

Čísla abundantní a deficientní

Co při výpočtu $s(n)$ vidíme? Pro některá n (např. $n = 12$) je $s(n) > n$, taková čísla se nazývají *abundantní*; pro jiná (např. $n = 10$) je $s(n) < n$, tato čísla se nazývají *deficientní*. V některých případech je $s(n) = n$, to jsou tzv. čísla *dokonalá*, což pro mnoho čtenářů jistě není novinkou.

Sledujme četnosti čísel abundantních a deficientních pro n v zadaném

rozmezí. K tomu čtenáři stačí jen mírně upravit předchozí program (vynechat tisk a zavést dvě počítadla), kterým pak četnosti lehce zjistíme. Zapišme si některé nahodilé výsledky do tabulky:

Rozmezí n	*abundantních*	*deficientních*
1 až 100 000	24 795	75 200
100 001 až 200 000	24 686	75 314
500 001 až 600 000	24 759	75 241
900 001 až 1 000 000	24 770	75 230
2 000 001 až 2 100 000	24 757	75 243
5 000 001 až 5 100 000	24 777	75 223

Takže se můžeme zeptat:

– Skutečně je v každém dostatečně rozsáhlém rozmezí n relativní četnost abundantních čísel mírně pod 25 % a deficientních mírně nad 75 %? (Na „přesnější“ úvahu máme zjištěno málo dat). A proč je tomu tak?

Dokonalá čísla

Právě tato čísla nejvíce znepokojují matematiky celých těch dva a půl tisíce let. Nebudeme se však zde jimi zabývat, jen si uveďme první čtyři: 6, 28, 496, 8128. (Nezasvěcený čtenář se může na internetu o těchto číslech a o historii jejich poznávání dozvědět mnoho zajímavostí, stačí vyhledat heslo „dokonalá čísla“; např. že všech 46 dosud známých dokonalých čísel jsou čísla sudá a že se dosud neví, zda existuje nějaké liché dokonalé číslo.) Mohli bychom mít námitku, proč právě vlastnost $s(n) = n$ je tak významná, že v ní spatřujeme jakousi *dokonalost*, ale berme to tak, že se ta čísla prostě takto *jmenují*.

Chce-li čtenář prožít pocit objevitele dokonalých čísel, může k tomu použít následující program, ale nesmí být zklamán, že těch dokonalých čísel moc neobjeví.

```
program DokCis;
var
  A, B, N: Longint;

function SD(M: Longint): Longint;
...
```

```
begin  {program}
  WriteLn('Dokonala cisla');
  Write('od A = ');
  ReadLn(A);
  Write('do B = ');
  ReadLn(B);
  N := ((A+1) div 2)*2;   {vytvoříme N sude}
  while N <= B do
  begin
    if SD(N) = N then WriteLn(N);
    Inc(N,2)
  end;
  WriteLn('Konec');
  ReadLn
end.   {program}
```

V programu na internetu je poznámka: „ ... liché dokonalé číslo asi neexistuje a pokud ano, tak je větší než 10^{300}, což je mimo rozsah formátu čísel i tohoto algoritmu“. Proto i v programu `DokCis` se postupuje jen po sudých číslech.

Posloupnosti navazujících čísel

Už pythagorejci zjistili existenci tzv. *spřátelených čísel*, což jsou takové dvojice n_1, n_2, pro něž platí $s(n_1) = n_2$ a současně $s(n_2) = n_1$, je to např. dvojice 220, 284. Takovými dvojicemi se zabýval např. i *L. Euler* a už se ví, že je jich nekonečně mnoho. (Pro čtenáře by nemělo být složité připravit si program, který by po spřátelených číslech pátral.) Jsou známa i tzv. *pospolitá čísla*, která jsou jakýmsi zobecněním spřátelených čísel a tvoří cyklus. Zatím byly objeveny jen čtyřčlenné skupiny, kde pro $i = 1, 2, 3$ platí $s(n_i) = n_{i+1}$ a dále $s(n_4) = n_1 \ldots$. Je to např. čtveřice 12 496, 14 288, 15 472, 14 536.

V tomto zobecňování pokračujme ještě dále a definujme *posloupnosti navazujících čísel*. Budeme tak označovat posloupnosti (konečné nebo nekonečné)

$$n_1, n_2, n_3, \ldots, n_k, \ldots,$$

v nichž pro všechna i platí $s(n_i) = n_{i+1} \ldots$. Takovou posloupností je např.

$$24, 36, 55.17, 1;$$

za poslední jedničkou by mohly následovat už jen samé jedničky, které však nebudeme uvažovat, a uvedenou posloupnost budeme považovat za konečnou. Jiný příklad:

$$222, 234, 312, 528, 960, 2\,088, 3\,762, 5\,598, 6\,570, 10\,746, \ldots;$$

její 108. člen je 1 677 601 896 (109. člen je již mimo možnosti formátu Longint) a je otázka, zda je tato posloupnost konečná nebo nekonečná, případně kolik má členů. Skutečnost, že se nám tato posloupnost jeví jako rostoucí, totiž ještě nic neznamená. Na počátku se takto stejně chová např. posloupnost.

$$120, 240, 504, 1\,056, 1\,968, 3\,240, 7\,650, 14\,112, 32\,571,$$

jejíž pokračování je

$$27\,333, 12\,161, 1,$$

neboť 12 161 je prvočíslo. Takže se nabízí otázka:

– Existuje nějaká nekonečná posloupnost navazujících čísel nebo jsou všechny tyto posloupnosti konečné?

Nabízíme čtenáři program, s jehož pomocí si může posloupnosti navazujících čísel pořizovat, a pak zkoumat jejich chování.

```
program PoNaCis;
var
  N, Suma: Longint;
  C: Integer;

function SD(M: Longint): Longint;
...

begin  {program}
  WriteLn('Posloupnost navazujicich cisel; prvni clen');
  Write('1.   ');
  ReadLn(N);
  C := 2; Suma := 0;
  while ((C <= 100) and (Suma <> 1)) do
  begin
    Suma := SD(N);
```

```
    WriteLn(C, '.   ', Suma);
    if (C mod 20) = 0 then ReadLn;
    Inc(C); N := Suma
  end;
  WriteLn('Konec');
  ReadLn
end.   {program}
```

Třídy $D(x)$

Nechť x je celé číslo. Třída $D(x)$ je množina všech takových přirozených čísel n, pro něž platí $s(n) = n + x$. Uvážíme-li, že $D(0)$ je množina všech dokonalých čísel, vidíme, že třídy $D(x)$ dávají na součty dělitelů obecnější pohled. Svůj název kdysi dostaly i prvky ze třídy $D(1)$, a to *skoro dokonalá čísla*, ale tato třída není až tak zajímavá, protože obsahuje právě jen všechny kladné mocniny dvou, tj. 2^k, kde k je přirozené číslo.

Vyšetřování tříd $D(x)$ je docela zajímavé, otevírá řadu problémů a umožňuje formulovat mnoho hypotéz. Uveďme příklady. V rozmezí n od 1 do 1 000 000 má třída $D(4)$ těchto 8 prvků:

$$12, 70, 88, 1\,888, 4\,030, 5\,830, 32\,128, 521\,726.$$

Ve stejném rozmezí má třída $D(-4)$ těchto 10 prvků:

$$3, 14, 44, 110, 152, 884, 2\,144, 8\,384, 18\,632, 116\,624,$$

zatímco např. třída $D(3)$ zde má jen jeden prvek, a to 18, a $D(3)$ dokonce žádný.

Podívejme se nyní v následujících tabulkách na počet prvků některých tříd $D(x)$ v rozmezí čísel n od 1 do 100 000 (první řádek) a od 1 do 1 000 000 (druhý řádek).

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
počet prvků	5	0	5	1	7	0	2	1	8	0	2	0	1 929	0	3	0
	5	0	7	1	8	0	3	1	11	0	3	0	15227	0	4	0

x	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
počet prvků	0	4	0	5	0	7	0	9	1	6	1	9	0	4	16
	0	4	0	6	1	9	0	12	1	6	1	10	0	4	19

Na základě poznatků z těchto tabulek nás jistě napadnou některé otázky, které mohou být podkladem různých hypotéz. Např.:

- Je to tak, že pro lichá $x \neq -1$ jsou příslušné třídy $D(x)$ prázdné nebo obsahují nanejvýš jeden prvek?
- Jak to, že třída $D(12)$ se počtem prvků tak prudce vymyká ostatním třídám? Má nekonečně mnoho prvků? Poznamenejme, že prvky třídy $D(12)$ nacházíme i mezi čísly velmi velikými (z hlediska fungování celých čísel na počítači), např. jich najdeme 8 v rozsahu od 2 147 483 000 do 2 147 483 600, tj. na samém okraji čísel formátu Longint. Největší dosažitelné našimi programy je 2 147 483 636.
- Vidíme, že pro námi zvolená sudá x se námi zjištěné prvky tříd $D(x)$ pro $x \neq 12$ nacházejí většinou nebo i všechny pro n již v rozmezí od 1 do 100 000 (případně ještě menším). Mají vůbec některé tyto třídy nekonečný počet prvků?

Je docela zajímavé vyzkoušet i další hodnoty x, dále od 0. Můžeme tak zjistit další anomálii, a to pro $x = 56$. Počet prvků $D(56)$ pro n v rozsahu do 100 000 je totiž 505 a v rozsahu do 1 000 000 je jich již 3 801.

- Proč pro $x = 56$ je četnost prvků $D(x)$ (v daném rozmezí n) výrazně vyšší? Jsou i další takové třídy $D(x)$ s vyšší četností prvků?

Uvedené počty prvků částí tříd $D(x)$ byly zjištěny tímto programem:

```
program TridyDX;
var
  A, B, N, Suma: Longint;
  Pocet, X: Integer;

function SD(M: Longint): Longint;
...
begin {program}
  WriteLn('Cisla z tridy D(x)');
  Write('x = ');
  ReadLn(X);
  Write('od A = ');
  ReadLn(A);
  Write('do B = ');
  ReadLn(B);
  Pocet := 0; N := A;
  while N <= B do
```

```
  begin
    Suma := SD(N);
    if Suma = N + X then
    begin
      WriteLn(N,'  ', Suma);
      Inc(Pocet)
    end;
    Inc(N)
  end;
  WriteLn('Pocet: ', Pocet);
  ReadLn
end.   {program}
```

Vraťme se ještě ke třídě $D(12)$ a vypišme si postupně od počátku několik jejích členů

$$24, 30, 42, 54, 66, 78, 90, \ldots;$$

zjistíme zajímavou věc, totiž že v $D(12)$ je obsažena pětičlenná podposloupnost navazujících čísel

$$30, 42, 54, 66, 78;$$

(ale dál už ne, protože $s(78) = 90 \notin D(12)$). Lehce se přesvědčíme, že navazujících *dvojic obsahuje* $D(12)$ celou řadu. Např. $246 \in D(12)$, $s(246) = 258 \in D(12)$; podobně namátkou vybereme dvojice 8 562, 8 574; 13 602, 13 614; 380 334, 380 346; 5 868 426, 5 868 438; 45 991 866, 45 991 878; atd., které mají stejnou výše uvedenou vlastnost. Možná, že se po přečtení informací o třídě $D(12)$ bude některému čtenáři jevit tato třída zajímavější a záhadnější než $D(0)$ – dokonalá čísla.

A přichází otázka:
- Kolik navazujících dvojic obsahuje $D(12)$? Obsahuje i jiné vícečlenné navazující podposloupnosti?
- Vyskytuje se tento jev i v jiných třídách $D(x)$?

Vyzkoušíme-li začátky námi zkoumaných tříd $D(x)$, objevíme jen jednu navazující dvojici ve třídě $D(10)$; $21 \in D(10)$, $s(21) = 11 \in D(10)$.

Třídy $Q(r)$

Nechť r je racionální číslo. Třída $Q(r)$ je množina všech takových přirozených čísel n, pro něž platí $s(n) = r \cdot n$. Vidíme, že $Q(1)$ je množina

všech dokonalých čísel, takže třídy $Q(r)$ také obohacují pohled na součty dělitelů. Chceme-li získat informace o „počátečních“ prvcích množin $Q(r)$, můžeme k tomu použít následující program.

```
program TridyQR;
var
  A, B, N, P, Q, Suma: Longint;
  Pocet: Integer;

function SD(M: Longint): Longint;
...

begin  {program}
  WriteLn('Cisla z tridy r(n)=(p/q)n');
  Write('citatel p = ');
  ReadLn(P);
  Write('jmenovatel q = ');
  ReadLn(Q);
  WriteLn('pro n v rozsahu');
  Write('od A = ');
  ReadLn(A);
  Write('do B = ');
  ReadLn(B);
  Pocet := 0;
  N := ((A + Q - 1) div Q) * Q;
  while N <= B do
  begin
    Suma := SD(N);
    if (Q * Suma = P * N) then
    begin
      WriteLn(N, '  ', Suma);
      Inc(Pocet)
    end;
    Inc(N,Q);
  end;
  WriteLn('Pocet: ', Pocet);
  WriteLn('Konec');
  ReadLn
end.   {program}
```

Pro představu o třídách $Q(r)$ si uveďme několik příkladů; pro n v rozmezí od 1 do 1 000 000 dostáváme pro zadaná r tyto prvky příslušných tříd:

$Q(3/2)$: 24; $Q(2/1)$: 120, 672, 523 746; $Q(5/2)$: 4 320, 4 680, 26 208; $Q(3/1)$: 30 240, 32 760; $Q(4/3)$: 12, 234; $Q(5/3)$; 84, 270, 1 488, 1 638, 24 384; $Q(7/3)$: 1 080, 6 048, 6 552, 435 708; $Q(3/4)$: 4; $Q(5/4)$: 40, 224, 174 592; $Q(7/4)$: 47 616.

Na základě získaných poznatků lze položit tyto otázky:
- Jsou všechny třídy $Q(r)$ konečné? Nebo je to jinak?
- Všechny zde uvedené příklady obsahují jen sudá čísla. Jak je to s lichými čísly, a existují i netriviální případy?

Triviálním případem přitom nazveme např. číslo 105 a jeho zatřídění: $s(105) = 87$, takže $105 \in Q(29/35)$.

Závěr

V tomto článku jsme položili několik otázek; jaké jsou možnosti najít na ně odpověď i s matematickým odůvodněním této odpovědi? Jsou v podstatě dvě:

- odpověď a její odůvodnění už matematika dávno zná, jen je třeba objevit příslušnou literaturu;
- odpověď v literatuře nenacházíme a pokoušíme se na základě hlubšího studia (k některé otázce) o vlastní závěr.

Věříme, že jsme tímto článkem inspirovali některé čtenáře, aby chtěli hlouběji proniknout do tajemství přirozených čísel a prožili přitom zajímavé chvíle.

Literatura

[1] *Redakce*: Rozklady složených čísel. MFI, 16 (2006–07), č. 1, str. 37–40.