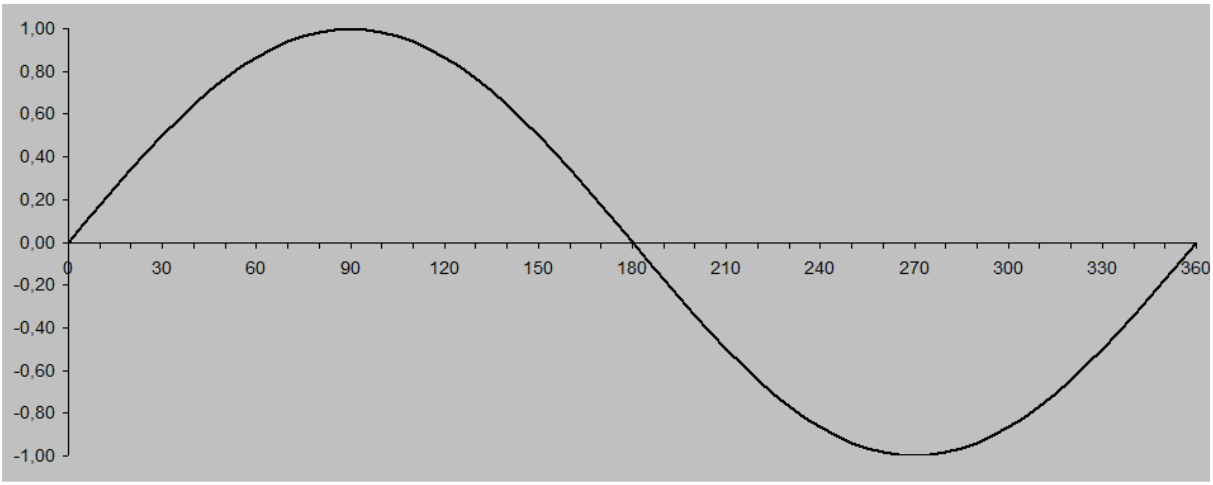
Zpravodaj moderníProgramování 05/2012: Jak se počítá sinus?

Předpoklad ke studiu tohoto čísla: učebnice pro začátečníky (modrá).

Co je sinus, se člověk učí na základní škole. Když jej potřebuje vypočítat, napíše do kalkulačky úhel a zmáčkne tlačítko „sin“. Když jej potřebuje vypočítat v programu (typicky pro různé úlohy s grafikou a kružnicemi, viz např. začátek žluté učebnice), zavolá metodu `Math.Sin` z knihoven .NET. Přemýšleli jste ale někdy, jak to vlastně ta kalkulačka nebo počítač vypočtou? Na to se podíváme v tomto čísle Zpravodaje. Náš výpočet bude zajímavý i proto, že si na něm procvičíme cykly, což není nikdy k zahození.

Výpočet sinu

Následující obrázek zachycuje známou „sinusovku“, neboli graf funkce sin. Přesněji řečeno, jednu její vlnu, která se periodicky po 360° opakuje vpravo i vlevo.



Jednou z možností, jak může počítač hodnotu sin x pro úhel x uvedený **v radiánech** vypočítat, je Taylorova řada, do níž lze rozvinout jakoukoli „rozumnou“ funkci. Pro funkci sinus vypadá následovně:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \ldots$$

Jde tedy o součet nekonečně mnoha členů, které se ale brzy začnou zmenšovat (a navíc střídají znaménko), takže prakticky stačí sečíst několik prvních členů.

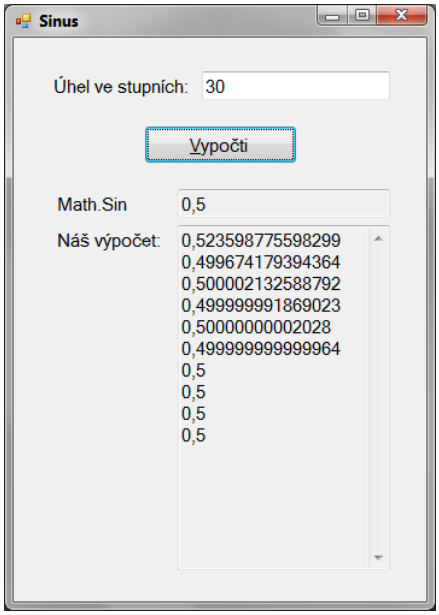
Pro připomenutí mocnina a faktoriál:

$$x^7 = x \cdot x \cdot x \cdot x \cdot x \cdot x \cdot x,$$

$$7! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7$$

Program

Vytvoříme program, který spočte součet prvních deseti členů Taylorovy řady, průběžné mezisoučty zobrazí a navíc pro kontrolu poskytne porovnání s výpočtem pomocí `Math.Sin`.



Zde je kód:

```
private void tlačítkoVypočti_Click(object sender, EventArgs e)
{
    // Vstupní údaj
    double úhelVeStupních = Convert.ToDouble(poleÚhel.Text);
    double úhelVRadiánech = úhelVeStupních * Math.PI / 180;

    // Knihovní výpočet
    double knihovníVýsledek = Math.Sin(úhelVRadiánech);
    poleKnihovníVýpočet.Text = knihovníVýsledek.ToString();

    // Náš výpočet - příprava
    poleNášVýpočet.Text = "";
    double x = úhelVRadiánech;
    double znaménko = 1;
    int n = 1;
    double součetŘady = 0;

    // Náš výpočet - jednotlivé členy řady
    for (int krok = 1; krok <= 10; krok++)
    {
        // Výpočet mocniny
        double mocnina = 1;
        for (int počet = 0; počet < n; počet++)
            mocnina *= x;
```

```
        // Výpočet faktoriálu
        double faktoriál = 1;
        for (int číslo = 2; číslo <= n; číslo++)
            faktoriál *= číslo;

        // Člen řady
        double členŘady = znaménko * mocnina / faktoriál;
        součetŘady += členŘady;
        poleNášVýpočet.Text += součetŘady.ToString() +
                               Environment.NewLine;

        // Příprava na další krok
        n += 2;
        znaménko = -znaménko;
    }
}
```

To nám to cyklí, co? Desetkrát počítáme x umocněno na příslušnou mocninu n (1, 3, 5, 7, ...) a současně $n!$ Po výpočtu člene jej přičteme k mezisoučtu, který zobrazíme. Mocniny jdou po dvou, znaménka se střídají.

Zrychlení

Program realizuje doslovný přepis vzorce Taylorovy řady pro sinus. Když se ale na vzorec podíváme podrobněji, můžeme si všimnout, že například:

- Když už jednou mám x^7, mám už skoro i x^9, proč tedy vlastně x^9 pracně počítat úplně znovu od začátku? Stačí vynásobit iksem a zase iksem...
- Když už jednou mám 7!, mám už skoro 9!, proč tedy 9! pracně počítat úplně znovu od začátku? Stačí vynásobit osmičkou a devítkou...

Jinými slovy, následující člen řady se mnohem rychleji vypočte z člene předchozího, než když jej počítáme pokaždé samostatně od začátku. V řadě programů na rychlosti tolik nezáleží, ale knihovní funkce pro sinus by určitě měla být co nejrychlejší, všichni ji budou používat, někdo možná mnohokrát v jednom výpočtu. V tomto případě tedy zrychlení stojí za to.

Rychlejší verze, která využívá zmíněné podobnosti členů řady, následuje:

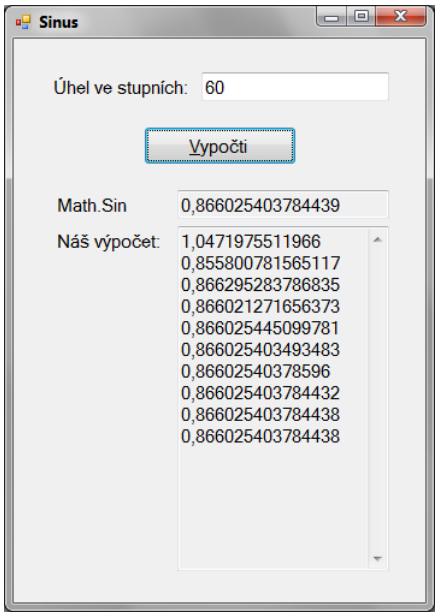
```
private void tlačítkoVypočti_Click(object sender, EventArgs e)
{
    // Vstupní údaj
    double úhelVeStupních = Convert.ToDouble(poleÚhel.Text);
    double úhelVRadiánech = úhelVeStupních * Math.PI / 180;

    // Knihovní výpočet
    double knihovníVýsledek = Math.Sin(úhelVRadiánech);
    poleKnihovníVýpočet.Text = knihovníVýsledek.ToString();

    // Náš výpočet - příprava
    poleNášVýpočet.Text = "";
    double x = úhelVRadiánech;
    double členŘady = x; // první člen
    int n = 1;
    double součetŘady = 0;
```

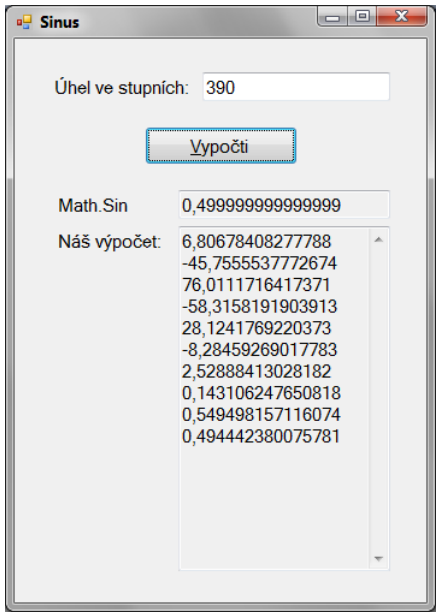
```
    // Náš výpočet - jednotlivé členy řady
    for (int krok = 1; krok <= 10; krok++)
    {
        // Přičteme aktuální člen řady
        součetŘady += členŘady;
        poleNášVýpočet.Text += součetŘady.ToString() +
                               Environment.NewLine;

        // Příprava na další krok
        n += 2;

        // TOTO JE HLAVNÍ ZMĚNA
        členŘady = -členŘady * x * x / (n * (n - 1));
    }
}
```

V tomto řešení zbytečně neopakujeme výpočty. Vše, co vypočítáme, použijeme dále.

Konvergence řady

Náš program pro ilustraci počítá a zobrazuje prvních deset mezisoučtů Taylorovy řady. Viděli jsme, že pro úhel 30° je již sedmý mezisoučet maximálně přesný. Čím ale větší úhel, tím je konvergence řady pomalejší (faktoriál ve jmenovateli tu mocninu v čitateli nakonec vždy udolá, ale pro velká x mu to trvá déle):





Vzniká tedy otázka, kolik členů řady by se mělo sečítat, jak velký zbytek řady lze již zahodit. Zde se opět využije matematika. Jestli si dobře pamatuji z analýzy, tak součet zbytku řady, jejíž členy střídají znaménka a v absolutní hodnotě klesají, nepřevýší první člen zbytku. Naše řada taková v zásadě je (alespoň pro malé úhly), takže by stačilo sčítat pomocí cyklu `while` tak dlouho, dokud člen řady neklesne pod požadovanou přesnost výpočtu.

Co se týče větších úhlů, určitě by bylo vhodné využít symetrie funkce, výpočet tak převést na malé úhly a tím zrychlit konvergenci. Pro sinus platí například:

- Periodicita: $\sin x = \sin (x + n \cdot 360)$
- Osová symetrie grafu v místě 90° neboli $\sin x = \sin (180 - x)$.
- Středová symetrie grafu v místě 0° neboli $\sin (-x) = -\sin x$

Závěr

Ukázali jsme si, jak asi je naprogramovaná metoda `Math.Sin`. Můžeme vidět, že hluboko v základech programování leží netriviální matematika. Především jsme si ale procvičili problematiku cyklů na trochu komplikovanější úloze. Viděli jsme také, jak vhodný trik může výpočet mnohonásobně zkrátit. No a v neposlední řadě jsme zkontrolovali Microsoft, že to asi naprogramoval správně!

Radek Vystavěl, 31. května 2012

Pokud Vám Zpravodaje moderníProgramování připadají užitečné, doporučte jejich odběr svým známým. Mohou se přihlásit na webu www.moderniProgramovani.cz.