Zpravodaj moderníProgramování 5/2013: Potřebujeme typ decimal?

Obtížnost: středně pokročilí

V C# existují tři datové typy pro ukládání desetinných čísel - `double`, `float` a `decimal`. Vztah prvních dvou je jednoduchý, `float` je prostě „malý `double`“. Na rozdíl od osmibytového `doublu` zabírá 4 byty, má menší rozsah a menší přesnost. Pokud ho nevyžaduje nějaká grafická funkce (třeba `DrawLine`...), není potřeba se s ním vůbec zabývat. Automaticky použijete `double` a je to. Mj. všechny matematické funkce vracejí `double`, takže tam není moc co řešit.

Typ `decimal` je ale jiný. Podívejme se, jak se liší od `doublu` a kdy má význam jej použít.

Základní odlišnosti

Typ `double`:

- Čísla jsou uložena ve dvojkové reprezentaci, což například znamená, že některá nevinná čísla jako třeba 0,8 mají **nekonečný periodický rozvoj**, tj. nelze je uložit přesně;
- Zabírá méně místa (8B proti 16B u `decimalu`);
- Výpočty jsou obvykle **mnohonásobně** rychlejší;
- Matematické funkce vracejí `double`.

Typ `decimal`:

- Čísla jsou uložena v desítkové reprezentaci, což znamená, že nevinně vypadající čísla jsou opravdu nevinná. Máte-li pár cifer za desetinnou čárkou, bude číslo uloženo přesně;
- Díky většímu místu, které využívá, má skoro extrémní přesnost (cca 28 platných cifer proti cca 15 u `doublu`).

Obvyklé využití typu `decimal` je pro **finanční výpočty**, kde se extrémně dbá na přesnost. Miliardové obraty musejí sedět na halíř. Že se přitom nějaký ten miliónek někam ulije, účetnictví neřeší. Čísla musí souhlasit do puntíku. Tento přístup tedy není všelék, ale určitě svůj význam má.

Pro ostatní druhy výpočtů (běžné výpočty, vědecko-technické výpočty) se použije známý `double`, k jehož komfortu přispívá to, že funkce třídy `Math` vracejí `double`.

Zmíněná extrémní přesnost se týká pouze nehmotných peněz, hmotných předmětů nikoli. Jestli ve skladu ubude mililitr benzínu, to se ani nedá změřit, mnohem více ho zůstane v hadici.

Práce s typem decimal

Typ `decimal` bohužel dělá problémy, protože není příliš slučitelný s typem `double`, jak ukazuje následující program:

```
private void OknoProgramu_Load(object sender, EventArgs e)
{
   // 1. Přímo uvedené hodnoty
   // double hodnota nemá příponu, nebo má příponu d
   double diblík = 1.2;
   // decimal hodnota má příponu m
   decimal decka = 3.4m;

   // 2. Přiřazení
   // 2a. Vzájemná přiřazení nejdou (jde o syntaktické chyby)
   // diblík = decka;
   // decka = diblík;
   // 2b. Nutno použít explicitní typovou konverzi
   diblík = (double)decka;
   decka = (decimal)diblík;

   // 3. Kombinace ve výpočtu
   // Musejí být buď všude doubly, nebo všude decimaly
   // Zkombinovat to nelze (jde o syntaktické chyby)
   // double výsledekDouble = diblík * decka;
   // decimal výsledekDecimal = diblík * decka;

   MessageBox.Show("Program nic viditelného nedělá");
   Close();
}
```

Přestože je, jak jsem uvedl, zvykem při práci s penězi používat typ `decimal`, je vzhledem k uvedeným těžkostem a mnohým výhodám `doublu` dobré zapřemýšlet, jestli se bez toho `decimalu` přece jen člověk neobejde.

Číselník

Určitě se bez něj neobejdete, když používáte ovládací prvek číselník (NumericUpDown). Jeho hlavní vlastnost `Value` má typ `decimal`, je tedy dost možné, že budete potřebovat buď výše uvedenou explicitní konverzi na `double` či `int`, eventuálně volání `Convert.ToDouble/ToInt32`.

Sólo hodnota

Dobře, číselník vzal čert, jak je to s těmi financemi? Jak může být velká hodnota, abychom ji v `doublu` uložili přesně na haléře? 15 platných cifer, minus 2 na halíře dává 13 cifer. To jsou jednotky bilionů. To nemá snad ani Kellner! Aspoň ne na jednom místě. Celková bilance banky? Státní dluh? Budiž. Pokud ale neprogramujeme pro banku, jednotlivá hodnota může být v `doublu` v pohodě, i kdybychom z těch 15 ještě něco málo obětovali jako zaokrouhlovací rezervu.

Sólo výpočet

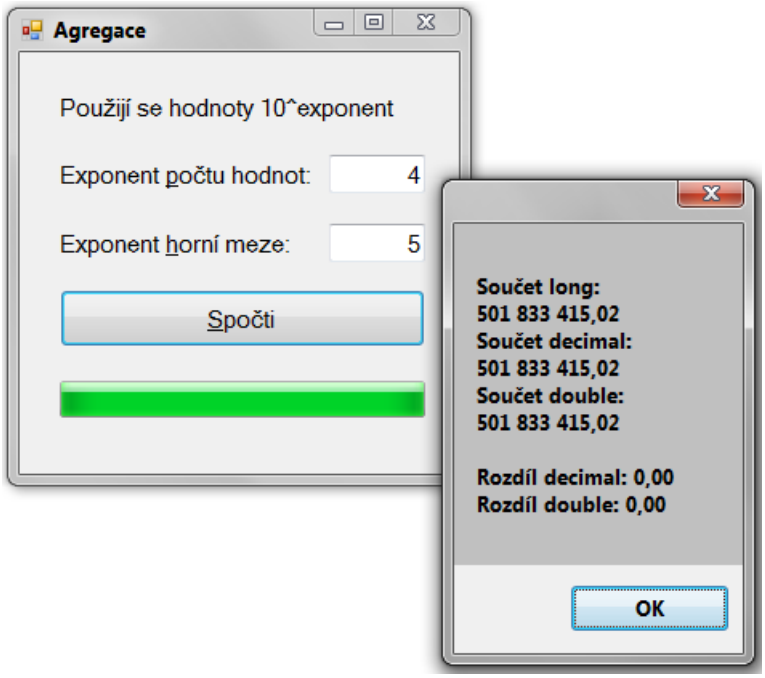
Co když s peněžní hodnotou provedeme jeden výpočet. Násobení velkým číslem asi není reálné - k čemu bude, když Kellnera vynásobím tisíci? Násobí se při úrocích, peníze krát nula celá něco (úrok), nebo krát jedna celá něco (nový zůstatek). Pořád je `double` v pohodě.

Samozřejmě, v ekonomicko-analytických výpočtech se mohou násobit dvě velká čísla, třeba při výpočtu korelačních koeficientů a podobně. V takových případech se ovšem na halíře nehraje.

Agregace

Kde můžeme narazit, jsou agregace. Mnoho hodnot sčítáme dohromady a pořád musí součet být správný na haléře. Tam můžeme narazit ještě dříve, než součet půjde do bilionů, neboť nedesítkově ukládané `doubly` mohou kumulovat zaokrouhlovací chybu. Jak je problém závažný, ověříme experimentálně. Sečteme mnoho náhodných hodnot v `doublu` a v `decimalu`. Pěknou kontrolu získáme sečtením haléřů v celých číslech (celá čísla jsou určitě přesná).

Uživatelské rozhraní:



Kód:

```
public partial class OknoProgramu : Form
{
    public OknoProgramu()
    {
        InitializeComponent();
    }
```

```
    private long Mocnina10(int exponent)
    {
        long výsledek = 1;
        for (int pořadí = 0; pořadí < exponent; pořadí++)
        {
            výsledek *= 10;
        }
        return výsledek;
    }

    private void tlačítkoSpočti_Click(object sender, EventArgs e)
    {
        // Vstupy
        int exponentPočtu, exponentHorníMeze;
        try
        {
          exponentPočtu     = Convert.ToInt32(poleExponentPočtu.Text);
          exponentHorníMeze = Convert.ToInt32(poleExponentHorníMeze.Text);
        }
        catch
        {
            MessageBox.Show("V obou polích nutno zadat celá čísla");
            return;
        }
        int exponentCelkem = exponentPočtu + exponentHorníMeze;
        if (exponentPočtu < 0 ||
            exponentHorníMeze < 1 ||
            exponentCelkem > 17 ||
            exponentPočtu == 0 && exponentCelkem > 16)
        {
            MessageBox.Show("Počet aspoň 0, mez aspoň 1, " +
                "jejich součet nejvýše 17 (16 pro exp. počtu 0)",
                "Nevhodné zadání");
            return;
        }

        // Příprava
        long součetLong = 0;
        double součetDouble = 0;
        decimal součetDecimal = 0;

        long početHodnot = Mocnina10(exponentPočtu);
        long krokUkazatele = početHodnot / 20 + 1;

        // Příprava náhody - částku vytvoříme ze tří čísel
        Random náhoda = new Random();
        int exponentVčetněHaléřů = exponentHorníMeze + 2;
        int exp1 = exponentVčetněHaléřů / 3;
        int exp3 = exponentVčetněHaléřů - 2 * exp1;
        int horníMez1 = (int)Mocnina10(exp1);
        int horníMez2 = horníMez1;
        int horníMez3 = (int)Mocnina10(exp3);
        long posun2 = horníMez1;
        long posun3 = posun2 * horníMez2;
```

```
        // Generování náhodných částek a sčítání
        for (long pořadí = 1; pořadí <= početHodnot; pořadí++)
        {
            // Tři čísla
            int číslo1 = náhoda.Next(0, horníMez1);
            int číslo2 = náhoda.Next(0, horníMez2);
            int číslo3 = náhoda.Next(0, horníMez3);

            // Náhodná částka v haléřích, resp. korunách
            long částkaLong = číslo1 + číslo2*posun2 + číslo3*posun3;
            double částkaDouble   = částkaLong / 100d;
            decimal částkaDecimal = částkaLong / 100m;

            // Přičítáme
            součetLong    += částkaLong;
            součetDouble  += částkaDouble;
            součetDecimal += částkaDecimal;

            // Ukazatel
            if (pořadí % krokUkazatele == 0)
            {
                ukazatelPostupu.Increment(5);
                ukazatelPostupu.Refresh();
            }
        }

        // Výsledky
        long koruny = součetLong / 100;
        long haléře = součetLong % 100;
        decimal součetLongKoruny = součetLong / 100m;
        decimal rozdílDecimal = součetDecimal - součetLongKoruny;
        decimal rozdílDouble = (decimal)součetDouble - součetLongKoruny;

        // Zobrazení výsledků
        ukazatelPostupu.Value = ukazatelPostupu.Maximum;
        ukazatelPostupu.Refresh();
        string zpráva =
            "Součet long: " + Environment.NewLine +
            koruny.ToString("N0") + "," + haléře.ToString("D2") +
            Environment.NewLine +
            "Součet decimal: " + Environment.NewLine +
            součetDecimal.ToString("N2") + Environment.NewLine +
            "Součet double: " + Environment.NewLine +
            součetDouble.ToString("N2") + Environment.NewLine +
            Environment.NewLine +
            "Rozdíl decimal: " + rozdílDecimal.ToString("N2") +
            Environment.NewLine +
            "Rozdíl double: " + rozdílDouble.ToString("N2");
        MessageBox.Show(zpráva);
        ukazatelPostupu.Value = ukazatelPostupu.Minimum;
    }
}
```

Poznámky k programu:

- Uživatel zadá počet náhodných částek a horní mez pro generování peněžní částky. Obojí zadává pomocí desítkového exponentu. Chceme-li milion čísel, zadáme šestku. Chceme-li částky maximálně devítimístné, tj. ve stamilionech, zadáme devítku (horní mez 10^9);
- Pro celočíselné výpočty použijeme 8-bytový `long`. Maximální hodnota, kterou lze v tomto typu uložit, je cca $9 \cdot 10^{18}$. Z toho vyplývají omezení na zadávané hodnoty;
- Náhodné částky se generují pomocí tří náhodných čísel, které se pak dávají dohromady;
- Program zobrazí součty v `longu`, v `decimalu` a v `doublu` a dále, o kolik se decimalové a doublové hodnoty liší od správného výsledku;
- Operace s takto velkými celými čísly jsou háklivé na celočíselné přetečení. Je třeba si minimálně hlídat, abychom nenásobili `int intem` - to by mohlo přetéct. Každopádně nemusí být od věci nastavit hlídání celočíselného přetečení. Jak se to dělá, popisuje jedenáctá kapitola učebnice pro začátečníky. Také je dobré pomocí ladicích prostředků se podívat do paměti, abychom se ujistili, že program opravdu pracuje, jak má.

Pokračování příště

Program dává velice pěkné výsledky. O těch si ale povíme až v dalším čísel Zpravodaje, kde se zaměříme i na rychlost výpočtů, no a potom vše shrneme do přehledného závěru. Zatím si můžete s programem experimentovat sami.

Radek Vystavěl, 7. srpna 2013

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