Číslo a název šablony	III/2 Inovace a zkvalitnění výuky prostřednictvím ICT
Číslo didaktického materiálu	EU-OPVK-VT-III/2-ŠR-106
Druh didaktického materiálu	DUM
Autor	RNDr. Václava Šrůtková
Jazyk	čeština
Téma sady didaktických materiálů	Programování v C# v příkladech I
Téma didaktického materiálu	Vlastní pomocné metody
Vyučovací předmět	Seminář z informatiky
Cílová skupina (ročník)	Žáci ve věku 16–17 let
Úroveň žáků	začátečníci
Časový rozsah	1–2 vyučovací hodiny
Klíčová slova	Běhové chyby, výjimky, typ double
Anotace	Studenti se seznamují s ošetřením běhových chyb pomocí mechanizmu výjimek a učí se pracovat s desetinnými čísly a metodami třídy Math.
Použité zdroje	DRÓZD, Januš a Rudolf KRYL. *Začínáme s programováním*. 1.vyd. Praha: Grada, 1992, 306 s. ISBN 80-854-2441-X. TÖPFEROVÁ, Dana a Pavel TÖPFER. *Sbírka úloh z programování*. Vyd. 1. Praha: Grada, 1992, 98 s. Educa '99. ISBN 80-854-2499-1. VYSTAVĚL, Radek. *Moderní programování: sbírka úloh k učebnici pro začátečníky*. 2. vyd. Ondřejov: moderníProgramování, 2008, 2 sv. ISBN 978-80-903951-5-2. VYSTAVĚL, Radek. *Moderní programování: učebnice pro začátečníky*. Ondřejov: moderníProgramování s.r.o, 2007, 2 sv. ISBN 978-80-903951-0-7.
Typy k metodickému postupu učitele, doporučené výukové metody, způsob hodnocení, typy k individualizované výuce apod.	Text je možno využít ke společné práci, samostatné přípravě studentů, domácímu studiu apod. Při společné práci je vhodné nejprve obtížnější úlohy rozebrat, potom společně se studenty implementovat na počítači. (Rozbor nejlépe na tabuli, synchronní řešení s promítáním) Prezentace obsahuje stručné shrnutí poznatků potřebných pro řešení příkladů. V pracovním listu je zadání cvičení – většinou se jedná o úlohy, které by měli studenti naprogramovat samostatně. Není nutné, aby všichni zpracovali všechno, vhodné je diferencovat podle jejich

	zájmu a schopností. Obtížnější úlohy jsou označeny hvězdičkou. Součástí materiálu je zdrojový kód těchto příkladů. Návrh způsobu hodnocení: ohodnocení samostatné práce během hodiny např. podle volby a počtu úloh a elaborace řešení (efektivnost, komentáře…).

Metodický list k didaktickému materiálu

Prohlášení autora

Tento materiál je originálním autorským dílem. K vytvoření tohoto didaktického materiálu nebyly použity žádné externí zdroje s výjimkou zdrojů citovaných v metodickém listu.

Obrázky (schémata a snímky obrazovek) pocházejí od autora.

106. Návratová hodnota, běhové chyby a výjimky, desetinná čísla

Návratová hodnota metod

Používáme-li zápis `x = Convert.ToInt32(textBox1.Text);` vlastně přiřazujeme do proměnné x celočíselnou hodnotu. Metoda `Convert.ToInt32(textBox1.Text)` tedy poskytuje hodnotu – hovoříme o její **návratové hodnotě**. (Je to jako když v matematice máme funkci y = f(x), pak např. f(3) je funkční hodnota v bodě 3 – v programování požíváme termín, že funkce vrací hodnotu f(3). Každá návratová hodnota je určitého typu – zde celé číslo. (ConvertToString(…) naopak poskytuje řetězec)

Jsou metody, které návratovou hodnot nemají – pouze představují akci: např. Close();

Zatímco metody s návratovou hodnotou najdeme např. na pravé straně přiřazovacího příkazu, metody bez ní tvoří samostatný příkaz. Uvádí se u nich pomyslný typ **void**. **(**prázdný)

Běhové chyby a jejich ošetření

Pokud do textového políčka, jehož obsah se má převést na číslo, zadáme něco, co číslo není, program přestane pracovat – říkáme, že došlo k **běhové chybě**. Program se vrací do vývojového prostředí, řádek, na kterém chyba vznikla, je žlutě označen. (Pád programu) Pokud by chyba vznikla mimo vývojové prostředí (spuštěním .exe souboru), program spadne rovněž, jen chybové hlášení vypadá jinak.

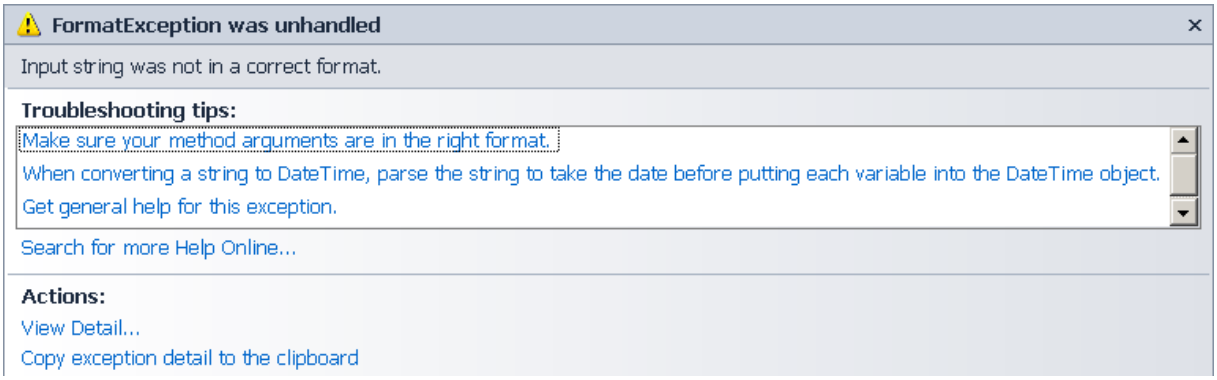
V programech určených pro uživatele, je třeba podobným chybám zabránit. Lze např. použít komponenty, které zadání chybného vstupu nedovolí, zde si ukážeme, jak zabránit pádu programu a přiměřeně upozornit uživatele na chybu pomocí tzv. **výjimky**.

Příklad 1

Napíšeme program pro výpočet druhé mocniny celého čísla. Číslo uživatel zadá do textBoxu (textBoxCislo) a po stisknutí tlačítka (buttonMocnina) by se měl zobrazit výsledek. do druhého textového pole (textBoxMocnina).

```
private void buttonMocnina_Click(object sender, EventArgs e)
        {
            int x = Convert.ToInt32(textBoxCislo.Text);
            int mocnina = x * x;
            //deklarace s inicializaci
            textBoxMocnina.Text = Convert.ToString(x) + " na druhou je " +
                Convert.ToString(mocnina);
        }
```

Pokud zadáme jako vstup číslo, je všechno v pořádku, pokud ovšem zadáme jiného, dojde k pádu programu a chybovému hlášení: – FormatException je tedy výjimka, která vznikne, když je vstupní řetězec nějak v nepořádku.



Pokud okno zavřeme, zůstane zvýrazněný řádek, na kterém došlo k chybě. Reagujeme přes nabídku **Debug/Stop Debugging.**

Postup: Označíme myší (nebo Shift + šipka) celý text mezi oběma složenými závorkami a z místní nabídky zvolíme **Surround With** a v dalším seznamu poklepeme na slovo **try.**

Zdrojový kód se obalí příkazy zachycení výjimky. Slova Exception a Throw vymažeme a Throw nahradíme vlastním chybovým hlášením:

```
private void buttonMocnina_Click(object sender, EventArgs e)
        {
            try
            {
                int x = Convert.ToInt32(textBoxCislo.Text);
```

```
            int mocnina = x * x;
            //deklarace s inicializaci
            textBoxMocnina.Text = Convert.ToString(x) + " na druhou je " +
                Convert.ToString(mocnina);
        }
        catch              {
          MessageBox.Show("Musíte zadat celé číslo");
        }
    }
```

Teď už program nepadá, ještě by bylo vhodné za příkaz `MessageBox.Show` přidat vymazání vstupu a předat vstupnímu políčku focus:

```
textBoxCislo.Text=null;
textBoxCislo.Focus();
```

Konstrukce **try-catch** uzavírá tzv. pokusný blok. Když vykonání některého příkazu v tomto bloku selže, místo běhové chyby se zbytek přeskočí a vykonají se příkazy v bloku za slovem **catch**. (**zachycení výjimky**)

Do pokusného bloku uzavíráme také všechny příkazy, které navazují na příkaz, který by mohl způsobit problém.

Desetinná čísla

Desetinná čísla se v počítači ukládají jinak než čísla celá a výpočty s nimi provádí jiná část procesoru. Tyto výpočty jsou náročnější a vzniká při nich zaokrouhlovací chyba. (Jednak do paměti vždy ukládáme jen určitý počet desetinných míst, jednak mohou chyby vznikat díky tomu, že počítač pracuje ve dvojkové soustavě.)

Pro desetinná čísla používáme typ **double** a převodní metodu **Convert.ToDouble**. Matematické funkce poskytuje třída **Math**.

π **Math.PI**

a^b **Math.Pow(a,b)**

$\sqrt{x}$ **Math.Sqrt(x)**

Zatímco ve vstupních políčkách zadáváme čísla s desetinnou čárkou (podle národního nastavení Windows), ve zdrojovém kódu se píše desetinná tečka. Podobně jako v Excelu se v matematických výrazech používají pouze kulaté závorky (dají se vnořovat) a priorita operací je stejná jako v matematice.

Aritmetika celých čísel je sice přesná, ale zase se do paměti nevejdou čísla libovolně velká – může docházet k přetečení. Proto vždy vhodně volíme datový typ podle úlohy.

Příklad 2

Vstupní údaj bude poloměr kružnice (textBoxR), program by měl vypočítat obvod a obsah kruhu, výsledky zobrazí do podobně pojmenovaných textových políček.

```
private void buttonKruh_Click(object sender, EventArgs e)
        {
            try
            {
                double r = Convert.ToDouble(textBoxR.Text);
                double O = Math.PI * r * r;
                double S = Math.PI * r * 2;
                textBoxObsah.Text = Convert.ToString(S);
                textBoxObvod.Text = Convert.ToString(O);
            }
            catch
            {
                MessageBox.Show("Musíte zadat reálné číslo");
                textBoxR.Text = null;
                textBoxR.Focus();
            }
        }
```

Důležité

Některé metody mají **návratovou hodnotu**, která se dosazuje za zápis volání metody.

Metody bet návratové hodnoty vystupují jako samostatný příkaz.

Běhové chyby a Výjimky: Příkazy, které by mohly způsobit problém, uzavíráme do pokusného bloku **try-catch**

Typ double používáme pro desetinná čísla, při práci s nimi může vzniknout zaokrouhlovací chyba.

Convert.ToDouble převádí na celé číslo.

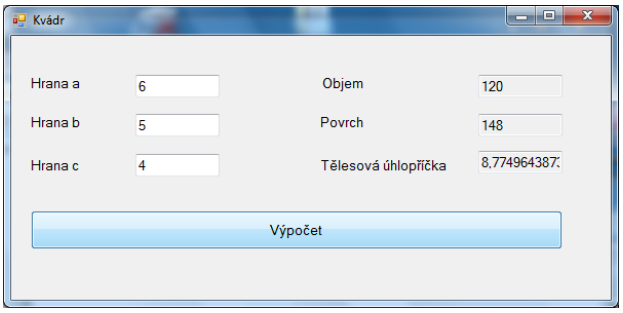
Ve Windows se desetinná část odděluje desetinnou čárkou, v kódu programu tečkou.

Při zápisu výrazů se užívají jen kulaté závorky.

Některé matematické funkce: Math.PI, Math.pow(zaklad,exponent) – mocnina, Math.Sqrt(x) – odmocnina

Pracovní list

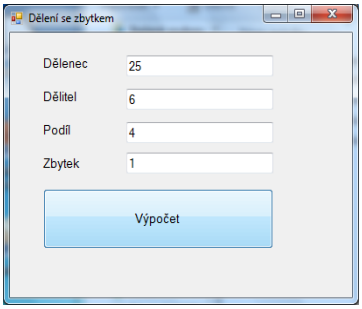
Cvičení



1. Připravte si program, který vypočítá objem, povrch a úhlopříčku kvádru. Ošetřete v něm výjimky.

2. Naprogramujte výpočet zbytku po celočíselném dělení podle vzoru deleni.exe. Ošetřete v něm výjimky.

(On na to je operátor %, který si můžete také vyzkoušet, ale vymyslete to i bez něj.)



3. Naprogramujte výpočet druhé odmocniny z reálného čísla. Ošetřete v něm výjimky.

Řešení

1.

```
private void buttonAkce_Click(object sender, EventArgs e)
        {
            try
            {
                double a = Convert.ToDouble(textBoxA.Text);
                double b = Convert.ToDouble(textBoxB.Text);
                double c = Convert.ToDouble(textBoxC.Text);
                //získání vstupních údajů z textových políček
                //a jejich převední na čísla
                double V = a * b * c;
                double S = 2 * (a * b + a * c + b * c);
                double u = Math.Sqrt(a * a + b * b + c * c);
                //výpočty podle vzorců
                textBoxS.Text = Convert.ToString(S);
                textBoxV.Text = Convert.ToString(V);
                textBoxU.Text = Convert.ToString(u);
```

```
            //zobrazení výsledků
        }
        catch
        {
            textBoxA.Text = null;
            textBoxA.Focus();
            textBoxB.Text = null;
            textBoxC.Text = null;
        }
```

2.

```
private void buttonAkce_Click(object sender, EventArgs e)
    {
        try
        {
            int delenec = Convert.ToInt32(textBoxDelenec.Text);
            int delitel = Convert.ToInt32(textBoxDelitel.Text);
            int podil = delenec / delitel;
            int zbytek = delenec - podil * delitel;
            textBoxPodil.Text = Convert.ToString(podil);
            textBoxZbytek.Text = Convert.ToString(zbytek);
        }
        catch
        {
            MessageBox.Show("Vstupem musí být přirozená čísla a dělitel
nesmí být nula");
            textBoxDelenec.Text = null;
            textBoxDelitel.Text = null;
            textBoxDelenec.Focus();
        }
    }
```

3.

```
private void button1_Click(object sender, EventArgs e)
```

```
        {
            try
            {
                double x = Convert.ToDouble(textBoxVstup.Text);
                double odmoc = Math.Sqrt(x);
                textBoxOdmocnina.Text = Convert.ToString(odmoc);
            }
            catch
            {
                MessageBox.Show("Musíte zadat kladné reálné číslo s desetinnou
čárkou!");
                textBoxVstup.Text = null;
                textBoxVstup.Focus();
            }
        }
    }
```