Střední škola pedagogická, hotelnictví a služeb, Litoměřice, příspěvková organizace

Předmět:	**Učební praxe**
Téma:	**Visual Studio**
Vyučující:	**Ing. Milan Káža**
Třída:	**EK3**
Hodina:	**27-30**
Číslo:	**V/5**

Programování v jazyce C a C#

Visual Studio

7. Program kalkulačka podruhé

Opět bude vytvořen program KALKULAČKA, ale bude provedeno ošetření pro vložení číslic, dále ošetření při dělení nulou a ukázka mocniny a odmocniny.

Jak vytvořit mocninu a odmocninu na druhou?

Jazyk C# má implementovanou matematickou knihovnu, ve které jsou různé matematické funkce a zde jsou některé z nich.

Min, Max: Min vrací nejmenší číslo a Max největší.

Round, Ceiling, Floor a Truncate: všechny tři funkce se týkají zaokrouhlování. **Round()** bere jako parametr desetinné číslo a vrací zaokrouhlené číslo typu **double** tak, jak je standartně (od 0.5 nahoru, jinak dolů). **Ceiling()** zaokrouhlí vždy nahoru a **floor()** vždy dolů. **Truncate** nezaokrouhluje, pouze odtrhne desetinnou část.

Abs a Sign: obě metody berou jako parametr číslo libovolného typu. Abs() vrátí jeho absolutní hodnotu a Sign() vrátí podle znaménka -1, 0 nebo 1 (pro záporné číslo, nulu a kladné číslo).

Sin, Cos, Tan: klasické goniometrické funkce, jako parametr berou úhel typu double, který považují v radiánech, nikoli ve stupních. Pro konverzi stupňů na radiány se stupně vynásobí *** (Math.PI/180)**. Výstupem je opět **double**.

Acos, Asin, Atan: opět klasické cyklometrické funkce (arkus funkce), které podle hodnoty goniometrické funkce vrátí daný úhel. Parametrem je hodnota v **double**, výstupem úhel v radiánech (také double). Pokud je potřeba mít úhel ve stupních, vydělí se **radiány / (180 / Math.PI).**

Pow a Sqrt: pow() bere dva parametry typu double, první je základ mocniny a druhý exponent. V případě výpočtu např. 2^3,

kód by byl následující:

```
Math.Pow(mocodmoc,expo);
```

Math je voláná matematická knihovna
Pow je funkce, která provede umocnění
mocodmoc je proměnná deklarováná jako **double**
expo je proměná která říká na kolikátou má být proměnná **mocodmoc** umocněná
celkový zápis vypadá takto:

```
expo = Int32.Parse(txbnantou.Text);
moc_vys= Math.Pow(mocodmoc,expo);
lb_vys.Text = "" + mocodmoc + " mocnina na" + "" + expo + "se rovná" + "" + moc_vys;
```

Poslední řádek, ukázaného kódu, zajistí výpis zadného čísla a na kolikátou bude umocněné a výsledek.

Další část vypsaného kódu zajišťuje aby byly vždy zapsány čísla a nikoli písmena, jedná se příkazy **try** a **catch.**

*Ve Visual Studia do Form pro Windows vložte dva **TextBoxy**, jeden **button** a **Label** pro výsledek, v prostředí pro tvorbu kódu vytvořte dvě proměnné typu double, ty pojmenujte **a, b** .*
Nyní napište následující příkazy a aktivujte tlačítko button a vyzkoušejte, pokud vložíte dvě čísla tak se vám zobrazí nápis
...

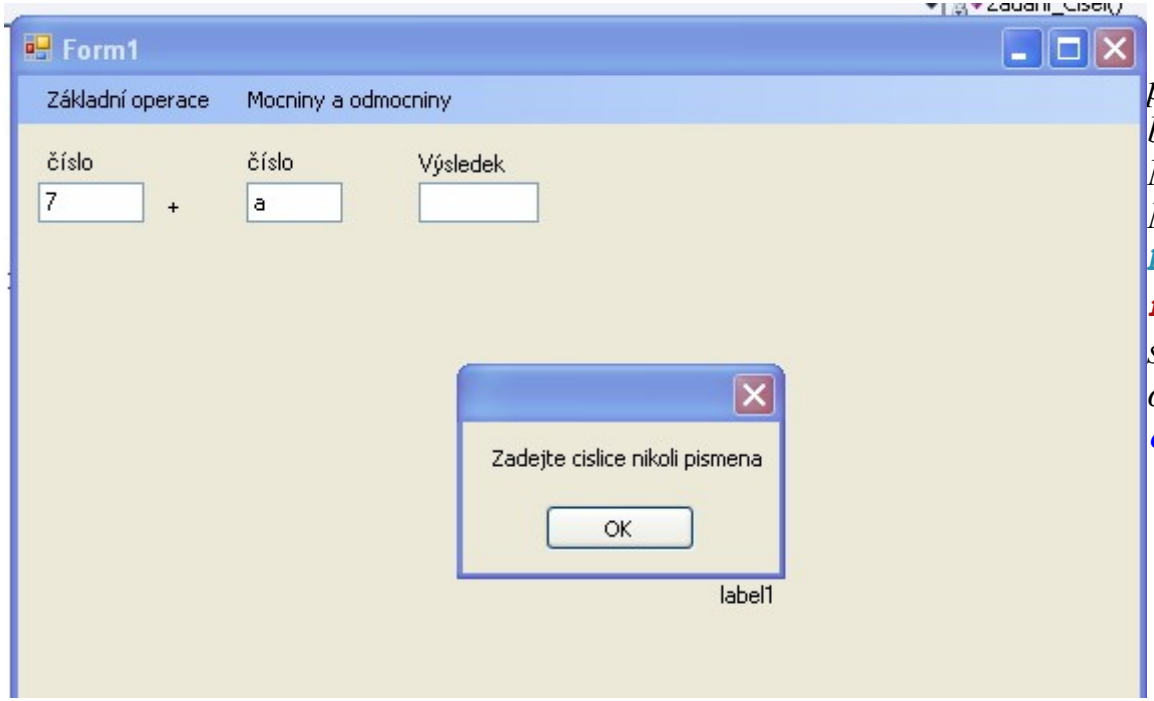
```
            try
        {
            a = Double.Parse(txb1.Text);
            b = Double.Parse(txb2.Text);
          Label.Text="Zadali jste číslo";
        }

    catch
```

```
{
    MessageBox.Show("Zadejte cislice nikoli pismena");

    ok = non_ok;
}
```

Pokud jste vložili jedno číslo a druhé písmeno tak se Vám zobrazí chybové hlášení.



Takto vypadá chybové hlášení, při zadání písmena místo číslice. Nápis si zvolíte sami, pokud budete programovat pro někoho vždy vykejte. Nápis chybovým hlášením vytvoříte příkazem MessageBox a zde je jeho clelá aplikace `MessageBox.Show("Zadejte cislice nikoli pismena");` *pro odstranění chyby stiskněte tlačítko OK a program se vrátí zpět a vy opravíte chyby. V tomto případě jste již v části* `catch.`

Vyzkoušejte si i další matematické funkce a zaveďte je do programu.
Vložte do Form Windows dva TextBoxy a jeden Button do progarmového kódu vložte proměnné jak je uvedeno v příkladech. Je možné vložit více Buttonů s nápisy podle uvedených funkcí.

Výpočet logaritmu

```
 double cislo1;
double zaklad;
double vysldek;
vysledek= Math.Log(cislo1,zaklad);
```

Převod z úhlu na radiany

```
double uhel;
double radiany;
radiany= (uhel*Math.PI)/100;
```

Zaokrouhlení číslice nahoru

```
double cislo;
double zaokrouhli;
zaokrouhli=Math.Ceiling(cislo);
```

Dále je možné provést výpočty úhlů získání absolutní hodnoty a výpočty mocnin a odmocnin.

7.1. Vytvoření kalkulačky – rozbor

Přestože každý z nás zná kalkulačku je vhodné si udělat před jejím vytvořením návrh jaké funkce kalkulačka bude mít a jak se budou používat.

Základní funkce: sčítání, odčítání, dělení a násobení.

Funkce sčítání

```
double cislo1;
double cislo2;
double vysledek;
vysledek=cislo1+cislo2;
```

Funkce odčítání

```
double cislo1;
double cislo2;
double vysledek;
vysledek=cislo1-cislo2;
```

Funkce násobení

```
double cislo1;
double cislo2;
double vysledek;
vysledek=cislo1*cislo2;
```

Funkce dělení

```
double cislo1;
double cislo2;
double vysledek;
vysledek=cislo1/cislo2;
```

U funkce dělění je nutné si uvědomit, že se nesmí dělit nulou a proto je nutné provést ošetření jedna z možností je tato

….

```
if (cislo2 == 0)
   {
   MessageBox.Show("Delit nulou nelze");
   }
    else
    { ….
```

V případě, že by chtěl někdo dělit nulou program nedovolí akci provést a oznámí, že nelze dělit nulou příkazem MessageBox.

Dále je požadavek na mocninu na druhou a vyšší a odmocninu.

Funkce mocnina na duhou

```
double mocodmoc;
double moc_vys;
moc_vys =  Math.Pow(mocodmoc,2);
```

Proměnná **mocodmoc** je pro zadání čísla a **moc_vys** je pro výsledek číslice 2 oznamuje, že číslice bude umocněná na druhou.

Funkce mocnina na duhou

```
double mocodmoc;
double moc_vys;
moc_vys =  Math.Sqrt(mocodmoc);
```

Sqrt je příkaz k provedení odmocniny ny druhou.

Funkce mocnina na n-tou

```
double mocodmoc;
double moc_vys;
int expo;
  expo = Int32.Parse(txbnantou.Text);
 moc_vys= Math.Pow(mocodmoc,expo);
```

Proměnná **expo** je definován jako integer zadává se pouze celé číslo.

Funkce odmocnina na n-tou

```
double mocodmoc;
double moc_vys;
int odmocexpo;
 odmcexpo = 1 / odmcexpo;
moc_vys= Math.Pow(mocodmoc, odmcexpo);
```

V případě odmocniny na n-tou se musí zavést příkaz **1/ odmocexpo** a poté umocnit výsledným číslem, jak je naznačeno na příkladu.

Program bude volat vždy danou matematickou funkci a využita bude pro to funkce **switch.**

```
private void Zakladni_Operace()
        {
            if (ok1 ==ok)
            {
                switch (promena)
                {
                    case 'a': vys = a + b;
                        txb3.Text = "" + vys;
                        lboperace.Text = "Operace sčítání";
                        break;
                    case 'm': vys = a - b;
                        txb3.Text = "" + vys;
                        break;
                    case 'k': vys = a * b;
                        txb3.Text = "" + vys;
                        break;

                    case 'd': if (b == 0)
                        {
                            MessageBox.Show("Delit nulou nelze");
                        }
                        else
                        {
                            vys = a / b;
                            txb3.Text = "" + vys;
```

```
                    }
                    break;
                                }
            }
            else
            {
                MessageBox.Show("Opravte vase chyby!!");
                ok = ok1;
            }
        }
```

```
private void btplus_Click(object sender, EventArgs e)
        {
         pnzaklop.Hide();
         Zadani_Cisel();
         Zakladni_Operace();
        }
```

Zde ještě bude objasněné volání funkcí. Funkce **privat void btplus_Click(..)** je funkce pro **Button**, kterou se zavolá funkce **private void Zakladni_Operace(),** v této funkci jsou zadané základní operace, sčítání, odčítání, násobení a dělení a současně je zde ošetření proti dělení nulou.

Ve starších aplikací se část, kam se odkazoval program nazývla podprogram, pokud bude použité toto označení není to chyba, v rámci programu je řešena určitá část.

Osobní zkušenost: Vytvořil jsem program, kde jsem potřeboval volat různé aplikace, každou jsem volal z hlavního programu a

výsledek každé části byl zpracován v další části. Osobně vidím výhodu v přehlednosti programu a jeho jednodušší údržbu, pokud vytvoříte aplikaci najednou těžko se upravuje a hledá se v ní těžko chyba. Je to můj osobní způsob každý programátor si najde časem svůj.

Kontrolní otázky

1. Jakým způsobem vložíte matematickou knihovnu.
2. Vysvětlete příkazy try a catch a objasněte jejich použití.

Použitá literatura:

Miroslav Virius: od C++ k C# ,1.vydání, KOOP České Budějovice, 2002
Eric Gunnerson: Začínáme programovat v C# , 1. vydání, Computer Press, 2001
John Sharp, Jon Jagger: Microsoft Visual C# .NET krok za krokem, 1. vydání, Knihy.iDnes
Miroslav Virius: C# Hotová řešení, 1. vydání, Computer Press, 2006
Amadeo Mareš: 1001 tipů a triků pro C#