Gymnázium Františka Palackého Valašské Meziříčí



Kalkulačka v C# .NET WPF

Seminární práce

Autorka: Anna Vičíková

Předmět: Programování

Třída: 4. B

Rok vypracování: 2017

Obsah

[Úvod 7](#_Toc510205526)

[1 Teorie Windows Presentation Foundation 9](#_Toc510205527)

[1.1 Co je to WPF? 9](#_Toc510205528)

[1.1.1 Komponentová architektura 9](#_Toc510205529)

[1.2 XAML 9](#_Toc510205530)

[1.2.1 Základní charakteristika XML a XAML 9](#_Toc510205531)

[1.2.2 Jmenné prostory 10](#_Toc510205532)

[1.2.3 Entity 11](#_Toc510205533)

[1.3 WPF a Windows Forms 11](#_Toc510205534)

[2 Vědecká kalkulačka 13](#_Toc510205535)

[2.1 Předmluva 13](#_Toc510205536)

[2.2 Vytvoření WPF aplikace 13](#_Toc510205537)

[2.2.1 První objekt 15](#_Toc510205538)

[2.2.2 Základní prvky kalkulačky 15](#_Toc510205539)

[2.2.3 Ošetření možných chyb, přiřazování kláves a přidání tlačítka pro desetinnou čárku 21](#_Toc510205540)

[2.2.4 Další operace 24](#_Toc510205541)

[2.2.5 Ošetření podruhé 27](#_Toc510205542)

[2.3 Grafická úprava WPF aplikace 27](#_Toc510205543)

[Závěr 29](#_Toc510205544)

[Seznam obrázků 31](#_Toc510205545)

[Reference 33](#_Toc510205546)

# Úvod

Tato seminární práce slouží jako návod, jak vytvořit vědeckou kalkulačku pomocí frameworku WPF (Windows Presentation Foundation) v programovacím jazyce C# a se samotným frameworkem se seznámit. Vybrala jsem si WPF, protože je současný a v dnešní době často využívaný.

# Teorie Windows Presentation Foundation

## Co je to WPF?

Windows Presentation Foundation je aplikační rámec (framework), který je součástí platformy .NET od verze 3.0. Aplikace založené na WPF se píšou kombinací jazyka C# a jazyka XAML (varianta jazyka XML, který umožňuje popis okna, jeho vlastností, ovládacích prvků v něm atd.). (1)

### Komponentová architektura

Framework se skládá ze spousty různých hotových komponent, ze který se následně formulář poskládá. Mluvíme tedy o tlačítkách, polích, popiscích, posuvnících a dalších, které Microsoft nazval controls (kontrolky). (2)

## XAML

**XAML = „XML for Application Markup Language“** (binární značkovací jazyk pro aplikace). Je to tedy značkovací jazyk XML pro aplikace. Popis okna v tomto jazyce se přeloží do binární podoby označované BAML (Binary Appliacation Markup Language), binární značkovací jazyk pro aplikace. V této podobě se připojí k výslednému sestavení jako prostředek (resource). (1)

### Základní charakteristika XML a XAML

XML se skládá z elementů. Struktura těchto elementů je stromová. Elementy tedy v sobě mohou obsahovat libovolné množství dalších elementů a tyto elementy další a další. Každý XML dokument obsahuje právě jeden kořenový element, ve kterém jsou dále umístěny podelementy. Vkládání elementů do sebe se nazývá **nesting** (hnízdění).

Elementy (mohou se nazývat také jako tagy, značky) se zapisují do lomených závorek, ve kterých je vždy název elementu. Každý element je nutné ukončit (na rozdíl od HTML, kde je ukončení nepovinné). Elementy ukončujeme podle toho, zda jsou **párové** nebo **nepárové**.

Nepárové elementy ukončujeme lomítkem před zavírací složenou závorkou:

<TextBox />

Párové elementy ukončíme umístěním další značky s názvem elementu, kde je po levé lomené závorce lomeno. Dovnitř párových elementů se vkládá obsah nebo další elementy. Obsah elementem označíme a přidělíme mu význam.

<TextBlock>Libovolný text</TextBlock>

Obsah můžeme vložit i do atributu, které se zapisují u párových a nepárových elementů stejně:

<TextBlock Text="Libovolný text" />

Atributů může mít každý element libovolné množství, protože jsou to vlastnosti daného elementu. Do atributu nelze vložit další element, ale pouze textovou hodnotu. (3)

### Jmenné prostory

Celkově v C# existují jmenné prostory, tzv. **balíčky, které v sobě obsahují jednotlivé třídy**, nebo další balíčky. Tyto balíčky byly uvedeny z toho důvodu, že se v .NET frameworku vyskytují opravdu obrovské množství tříd (ve WPF obsahují elementy). Bez těchto jmenných prostorů by musela každá třída mít unikátní název a taktéž bychom nemohli vytvořit vlastní třídu, která se jmenuje jako nějaká třída z .NETu. Tento problém je nazýván jako **kolize názvů**. (3)

Jmenné prostory využíváme právě tehdy, když si je „vyvoláme“ na začátku dokumentu. Musíme uvést které komponenty z kterých jmenných prostorů budeme využívat. Můžeme tak mít několik komponent se stejným názvem, pokud používáme v jednu chvíli jen jednu z jednoho jmenného prostoru. Pomocí tzv. **aliasů** můžeme komponentu pro daný dokument dospecifikovat a tak používat více komponent se stejným názvem v jednu chvíli pod jiným označením. (3)

### Entity

Když budeme chtít do obsahu elementu nebo atributu vložit nějaký znak (závorka, uvozovka, lomítko atd.), budeme mít jistě problém s tím, že takové znaky se ve WPF používají v syntaxi. Proto byly zavedeny entity, což jsou náhrady zejména za znaky <, >, &, “. (3)

|  |  |  |
| --- | --- | --- |
| Zápis | Název | Znak |
| &lt | less than (méně než) | < |
| &gt | greater than (větší než) | > |
| &amp | ampersand | & |
| &quot | quotation mark (uvozovka) | “ |

## WPF a Windows Forms

Windows Forms je prvním frameworkem z .NETu, který slouží k tvorbě jednoduchých formulářových aplikací pomocí grafického designeru. V dnešní době už ho nahrazuje mnohem modernější framework WPF, který obsahuje lepší oddělení logiky a výstupu, podporu rychlejšího vykreslování, animací, bindingů atd. V praxi se využívají oba frameworky, ale WPF je technologicky mnohem dále. (2)

Ke vzniku nového frameworku došlo kvůli mnoha nevýhodám Windows Forms. Například, když začaly dominovat na trhu mobilní zařízení, WF aplikace začaly selhávat ohledně uzpůsobení jejich fyzické velikosti kvůli slabé podpoře DPI. Proto nemůžeme používat jednu a tu samou aplikaci např. na mobilu, tabletu a stolním počítači s full HD rozlišením. WPF zavádí jednotku délky, DIP (Device Independent Pixel) a čistě vektorovou grafiku, aby přizpůsobila své rozlišení na jakémkoliv zařízení. (2)

Další nevýhodou WF je ukládání **absolutní pozice** na formuláři u každého ovládacího prvku, které se pro návrh složitějších formulářů nehodí. Jak je v případě Javy, která se zavádí definici formulářů pomocí jazyka XML, tak se inspiroval C# pomocí XAML. Tento jazyk nám přináší lepší oddělení prezentace a logiky, jednoduchou podporu tzv. **bindingů**, které umožňují napojit vlastnosti objektů přímo na ovládací prvky formuláře. (2)

Zatímco WF využívá původního vykreslovacího rozhrání Windows (GDI), které je velmi pomalé, WPF využívá k vykreslování formulářů rozhrání pro akcelerovanou grafiku **Direct3D**. Aplikace jsou rychlejší a méně zatěžují procesor. Díky nezávislosti na GDI je možné se odpoutat od palet základních ovládacích prvků a vytvářet graficky bohaté aplikace, protože zde nejsou žádná omezení. Můžeme např. vkládat obrázky do tlačítek, do položek comboboxu, cokoli zprůhlednit, skinovat naše aplikace nebo dokonce animovat pomocí tzv. storyboardů. (2)

# Vědecká kalkulačka

## Předmluva

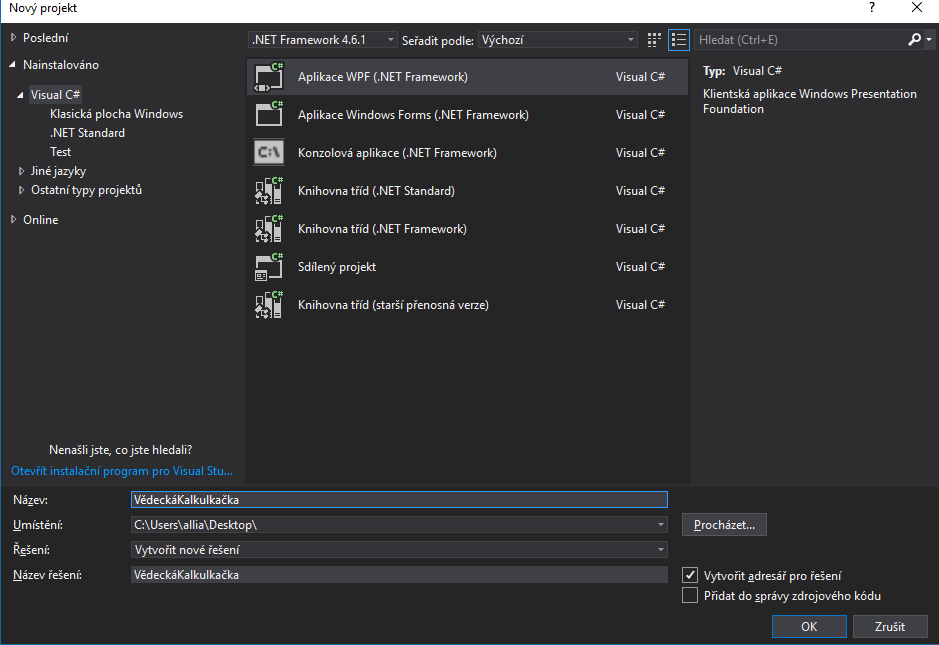
Než se pustíme do samotného programování, je důležité rozmyslet si, co vše naše kalkulačka bude obsahovat a umět. Bude umět základní operace, tzn. sčítat, odčítat, dělit a násobit? Dokázala by vypočítat druhou mocninu libovolného čísla? Logaritmovat libovolné číslo? Dokázala by vypočítat goniometrické funkce?

Poté si položme otázku, když uvažujeme nad složitými operacemi naší budoucí kalkulačky, dokážeme něco takového naprogramovat?

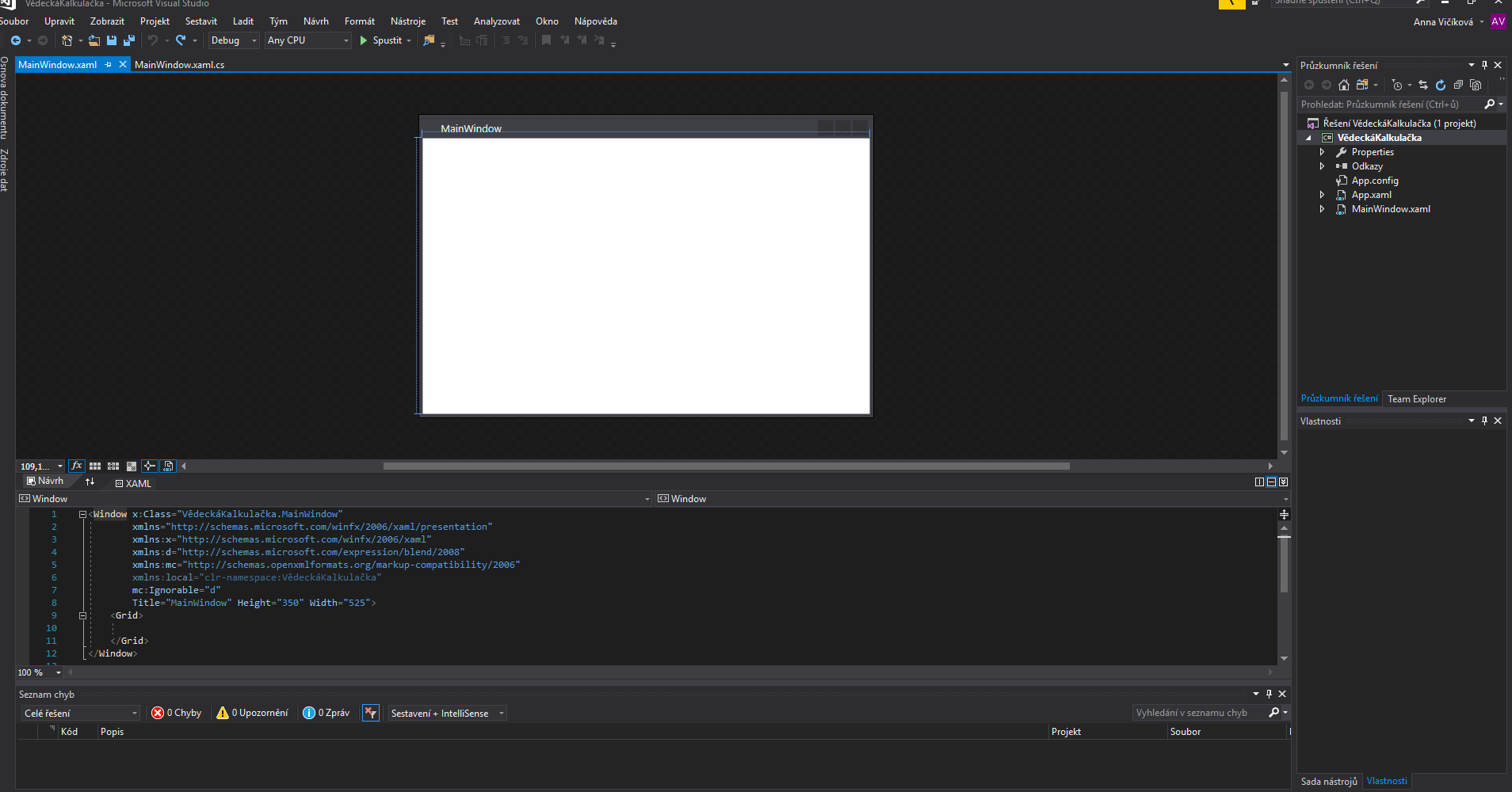
Jednoznačná odpověď je ano. Dle mého názoru, každý dokáže naprogramovat složité funkce, jen si k tomu musí najít cestu. Stejně tak my, programátoři, půjdeme krůček po krůčku při programování kalkulačky. Nejdříve však je důležité se seznámit s celým prostředím.

## Vytvoření WPF aplikace

Ve Visual Studiu si vytvoříme nový projekt. Nazveme ho třeba „VědeckáKalkulačka“ a zvolíme si „**Aplikace WPF (.NET framework)“**.



Obrázek 1 Vytváření nového projektu ve Visual Studiu



Obrázek 2 Rozhrání aplikace ve Visual Studiu

Zobrazila se nám hromada panelů. Veprostřed je zobrazený **grafický designer**, tzn. jak bude vypadat náš program. Pod ním se vyskytuje **XAML kód**, který popisuje vzhled formuláře. V pravé části pod průzkumníkem máme **panel vlastností** kontrolek (ve formuláři zatím nemáme žádnou kontrolku, proto je panel prázdný) a dole po rozkliknutí **panelu nástrojů** se nám ukážou všechny možné již navrhnuté kontrolky, které můžeme využít.

### První objekt

Zkusíme do formuláře přidat náš první objekt, abychom si vyzkoušeli, jak takový objekt vypadá a jak se s ním pracuje.

#### První způsob

Podívejme se na Button (tlačítko). Najdeme ho v panelu nástrojů a jednoduše ho přetáhneme do formuláře. Tím se nám ve formuláři ukázalo tlačítko a návrhář pro něj vygeneroval kód.

#### Druhý a efektivnější způsob

Vždy existuje mnohem lepší způsob, než aby nám kód vygeneroval počítač, a to ten, že si ho napíšeme sami. Jak vypadá takový kód pro Button, už víme z předešlého generovaného kódu, stačí si zapamatovat strukturu a můžeme si vytvořit jakýkoliv objekt (tím už vlastně ovládáme jazyk XAML).

Proč bychom si měli psát vlastní kód, když je mnohem jednodušší a rychlejší, když se sám vygeneruje?

Kdybychom měli přesně nadefinovat, jak a kam chceme umístit kontrolku, návrhář by nám mohl vygenerovat dost složitý a neefektivní kód. Proto je mnohem lepší varianta si to sám napsat.

### Základní prvky kalkulačky

Do naší kalkulačky budeme moct jak vypisovat čísla a operace z klávesnice, tak je i zadávat přímo v kalkulačce. Program bude mít tím pádem 10 tlačítek pro jednotlivá čísla (0–9) a 4 tlačítka pro základní operace (+,–,\*, /), jedno tlačítko pro výsledek a prostor pro vstup/výstup (Textbox)

#### Návrh kalkulačky

Dříve než si vytvoříme tlačítka, tak si uděláme návrh kalkulačky. Pro návrh využijeme **Grid**, vnitřní část formuláře. Když klikneme doprostřed formuláře, označí se a my si ho můžeme libovolně rozdělit do tabulky. Velikost jednotlivých sloupců a řádků můžeme následně i změnit v kódu, který návrhář generuje. Já vytvořila 5 sloupců o rozměru 104 a 6 řádků, z nichž jeden o rozměru 70 (zde bude prostor pro zadávání a výpočet) a ostatní o rozměru 50. Když budeme vkládat objekty do jednotlivých rozdělených dílků, jsou tady vlastnosti **Grid.Row** (číslo řádku) a **Grid.Column** (číslo sloupce). Čísla se počítají od levého horního „rohu“ a to od 0.

#### Vložení tlačítek

Nyní si všechny prvky napíšeme do **Gridu** za konec definice řádků a sloupců. Tlačítko pro číslo 0:

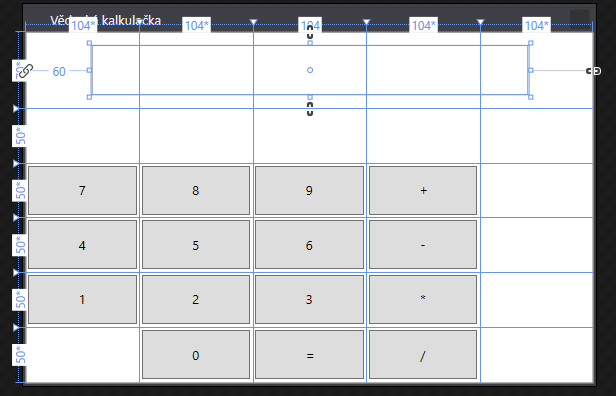
<Button x:Name="C0\_button" Content="0" Height="45" Width="99" VerticalAlignment="Center" Grid.Row="5" Grid.Column="1" HorizontalAlignment="Center" Click="C0\_button\_Click" />

Stejně můžeme vytvořit ostatní tlačítka.

#### Vložení Input/Output

Jelikož víme, že Textbox by měl zaplňovat téměř celou šířku formuláře, **Grid.Column** nám nebude stačit, protože to znamená, že by byl pouze v tom malém dílku. Proto využijeme vlastnost **Grid.ColmunSpan**, která nám zajistí, aby se posunul do středu daného dílku, ale šířka by přesahovala dílky ostatní.

<TextBox x:Name="hlavni\_textbox" Height="45" VerticalAlignment="Center" Grid.Row="0" Grid.ColumnSpan="5" Margin="60,12,0,12" HorizontalAlignment="left" Width="400" FontSize="20"/>



Obrázek 3 Rozdělení okna a přidání objektů

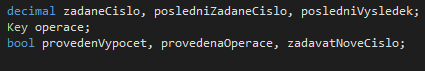
#### Základní operace

Jak už jsem se dříve zmínila, měli bychom jít pomalu a postupně. Dříve, než vytvoříme ostatní operace, naprogramujeme ty základní. K čemu by nám byly např. goniometrické funkce, když by kalkulačka neuměla sčítat ani odčítat?

Pojmenujeme si všechny buttony, např. „c0\_button“, kde c značí číslo, 0 jako nulu a button, abychom nezapomněli, s jakým prvkem pracujeme. Doporučuji to tak udělat u všeho, aby se nevyskytl zmatek v průběhu programování.

Přejdeme do interakční logiky **„MainWindow.xaml.cs“**, která je téměř prázdná. Zde máme prostor pro programování jednotlivých operací.

Nejdříve si nadefinujeme prvky, které budeme používat ve všech operátorech, a to jsou čísla a operace. Tyhle prvky se definují v třídě MainWindow.

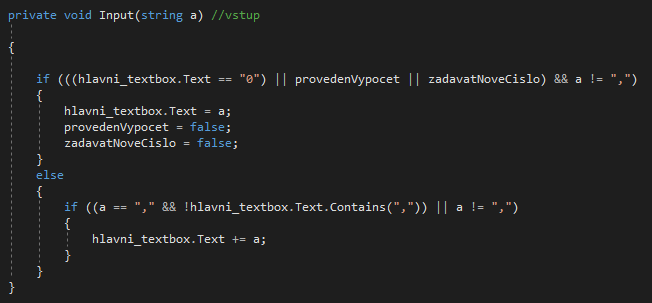


Obrázek 4 Deklarace globálních proměnných

Je důležité brát ohled, v jakém datovém typu definujeme čísla, protože každý datový typ má různý rozsah. Například, datový typ **float** má rozsah ±1.5 \* 10−45 až +-3.4 \* 1038, na rozdíl od něj datový typ **double** má rozsah +-5.0 \* 10−324 až +-1.7 \* 10308, takže co se týkávelikosti, vyhrává double. Poté ještě existuje datový typ **decimal**, který má rozsah o dost menší než double a **float** (rozsah +-1.0 \* 10−28 až +-7.9 \* 1028), tím pádem nezabírá ani tolik paměti, takže je pro naši kalkulačku nejvíce vhodný.

Nedefinujeme pouze jedno číslo, ale definujeme dvě čísla a výsledek (kvůli ošetření). Klíč u operace ošetřuje tlačítko na klávesnici (určitě chceme, aby náš program byl ovládaný nejen klikáním, ale i z klávesnice). Poté pro ošetření jsme nadefinovali datovým typem **bool**, jestli byl proveden výpočet a jestli byla provedena operace.

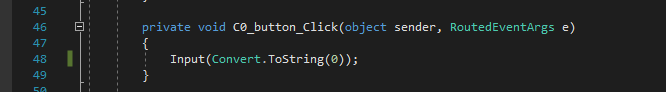
Po deklaraci prvků, které budeme po celou dobu používat, vytvoříme metodu Input. Tato metoda nám bude vypisovat čísla, která naklikáme, do Textboxu.



Obrázek 5 Kód pro Input

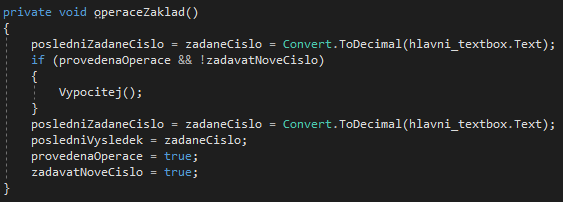
Pokud v textu Textboxu bude 0, tak se toto číslo přepíše na číslo a (když budeme zadávat první číslo), nebo když už bude proveden nějaký výpočet (abychom mohli dále pracovat s výsledkem dále), nebo když se bude zadávat nové číslo. Pokud už tam nějaké číslo bude, tak se k němu přidá další číslo, které zadáme. Tato metoda je nutná při zadávání čísel a operací přímo v programu, ale také i z klávesnice. Zároveň jsme ošetřili první problém s čárkou. Pokud se čárka v Textboxu bude vyskytovat, při zmáčknutí klávesy (kterou si později naprogramujeme), se nám čárka nenapíše znovu.

Tuhle metodu přiřadíme ke každému číslu, např.



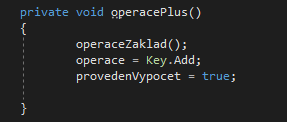
Obrázek 6 Přiřazení čísel k tlačítkům

kde a v metodě je dané číslo.

Nyní naprogramujeme operace. Bude existovat jedna metoda operace, která bude platit pro všechny základní operace. Nazveme si jí **„operaceZaklad“,** kde přiřadíme poslední zadané číslo*,* a pokud ještě nebyla provedená operace a zároveň se zadávalo nové číslo, kalkulačka vypočítá tuto událost. 

Obrázek 7 Metoda pro základní operace

Poté vytvoříme metody pro jednotlivé základní operace, ke kterým přiřadíme základní metodu a jejich klávesy. Stisknutí této operace rozumíme zmáčknutí klávesy na klávesnici. Na klávesu „**+**“ na klávesnici nás navede **Key.Add**, pro „-“ nás navede **Key.Substract**, pro „\*“ je **Key.Multiply** a „/“ je **Key.Divide**.



Obrázek 8 Příklad metody pro operaci sčítání

Takhle jsme si vytvořili metodu pro operaci sčítání. Podobné to tedy bude i pro odčítání, násobení i dělení.

Nyní se přesuneme na metodu, která nám vše vypočítá.

V této metodě opět nadefinujeme poslední zadané číslo (pokud již byl proveden výpočet) a nastavíme **bool** u provedení výpočtu na **true** (protože po použití této metody výpočet nastane) a **bool** u provedené operace na **false** (ten se nastaví po stisknutí dané operace). Pomocí switch si vybereme operaci a ošetříme dělení nulou.Obsah obrázku snímek obrazovky

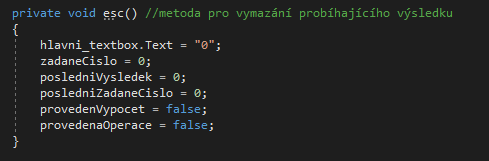
Popis vygenerován s velmi vysokou mírou spolehlivosti

Obrázek 9 Metoda pro výpočet

Poté tuto metodu přiřadíme k tlačítku pro výsledek.

#### Tlačítko C

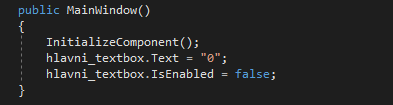
Vytvoříme tlačítko C pro vymazání probíhajícího výsledku. Tomuto tlačítku přiřadíme její *(jeho)* metodu, která vynuluje všechny proměnné.



Obrázek 10 Metoda pro tlačítko C

### Ošetření možných chyb, přiřazování kláves a přidání tlačítka pro desetinnou čárku

Aby náš program pořádně fungoval, musíme ho taktéž i pořádně ošetřit od případných errorů a shazování programu.Pár ošetření už jsme provedli během programování základních operací. Nejprve ošetříme klávesy. Do našeho programu můžeme totiž libovolně psát jak čísla, tak i písmena, se kterýma nám program po kliknutí operace spadne. Proto zakážeme, abychom mohli psát do Textboxu.



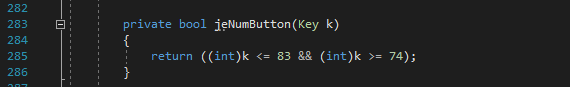
Obrázek 11 Zákaz psaní do Textboxu v okně

Abychom mohli zadávat čísla a operace z klávesnice, vytvoříme metodu na zjištění klávesy. Když se zjistí, jakou klávesu jsme stlačili, tak se vykoná metoda k dané klávese (v případě, že to je klávesa pro operaci) nebo se vepíše číslo.

Obsah obrázku snímek obrazovky

Popis vygenerován s velmi vysokou mírou spolehlivosti

Obrázek 12 Metoda pro zjištění klávesy



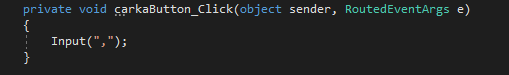
Obrázek 13 Metoda pro zjištění klávesy na numerické klávesnici

Tak jsme si pojistili všechny klávesy na numerické klávesnici, operace a klávesy E a D. Přiřadíme klávesu i pro tlačítko C – tomu odpovídá klávesa ESC. V běžné kalkulačce se vyskytuje i tlačítko CE, která vymaže zobrazené číslo. Dle mého názoru si vystačíme pouze s klávesou D.

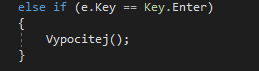
Pro využití metody se v XAML prostoru dopisuje do Window příkaz **Keydown**.

KeyDown="ZjistiKlavesu"

Protože pracujeme s čísly, která nemusí být celá, budeme jistě používat i desetinnou čárku. Do programu vložíme nové tlačítko s čárkou a nadefinujeme. Také nadefinujeme i klávesu na klávesnici (do metody pro zjištění klávesy)



Obrázek 14 Tlačítko pro desetinnou čárku



Obrázek 15 Přiřazení klávesy Enter pro výpočet

Při zmáčknutí tlačítka pro čárkuse do Textboxu přidá čárka. Problém, kdy by se čárka zadávala znovu, je již ošetřený, když jsme vytvářeli Input.  
Když si vyzkoušíme náš program, tak při zadávání více příkladů z klávesnice se jako výsledek vyhodnotí nula. Je to tím, že se program po prvním výpočtu už nadále do metody Vypocitej nedostane, kvůli tomu, že program „pozastavil“ při zadávání čísel. Při zadávání čísel se program zaměří na poslední stisknuté tlačítko. Tento problém nám vyřeší vlastnost **Focusable**, kterou nastavíme na false u každého tlačítka.

#### Tlačítko backspace

Když napíšeme nějaké číslo a budeme ho chtít vymazat, použijeme klávesu backspace nebo tlačítko na kalkulačce. Když zmáčkneme klávesu, vymaže se poslední číslo v Textboxu (pokud to není nula), ale v případě, že je to jediné číslo, vrátí nám to nulu. Vytvoříme si pro tento úkon metodu, protože ji využijeme jak u klávesy na klávesnici, tak i u tlačítka na kalkulačce.

private void backspace()

{

if (!String.IsNullOrEmpty(hlavni\_textbox.Text))

{

hlavni\_textbox.Text = (String.IsNullOrEmpty(hlavni\_textbox.Text.Remove(hlavni\_textbox.Text.Length - 1))) ? "0" : hlavni\_textbox.Text.Remove(hlavni\_textbox.Text.Length - 1);

}

else

{

hlavni\_textbox.Text = "0";

}

}

Pro lepší vzhled kalkulačky přiřadíme tlačítku backspace znak „Erase to the left“ (u vlastnosti Content). (3)

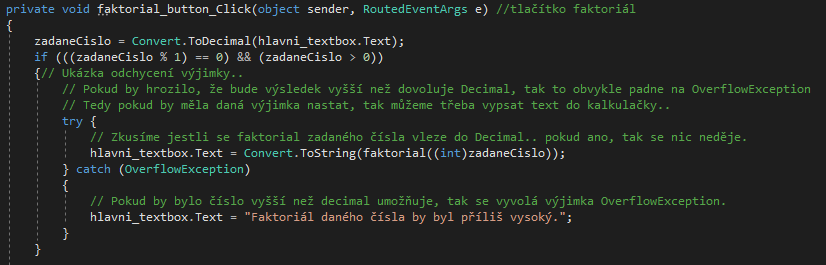
Obsah obrázku snímek obrazovky

Popis vygenerován s velmi vysokou mírou spolehlivosti

Obrázek 16 Tlačítko backspace se znakem Erase to the left

### Další operace

#### Faktoriál

Faktoriál daného čísla můžeme naprogramovat více způsoby (cyklem for, while,…), nejúčinnější cesta je rekurzí (kdy metoda volá sama sebe). Faktoriál čísla n je číslo rovné součinu všech kladných celých čísel menších nebo rovných n. (4) Musíme zajistit, aby zadané číslo bylo tedy kladné celé číslo. To lze ošetřit také mnoha způsoby (**int.TryParse**, pomocí **boolean**,…), nejjednodušší je dělení beze zbytku. Když vydělíme číslo jedničkou a nic nám nezbude, je to celé číslo. Mějme na paměti, že se stále vyskytujeme v datovém typu Decimal, proto jsme docela dost omezeni. S faktoriálem se můžeme dostat do vysokých čísel a je důležité odchytit výjimku (**OverflowException**). Tuto výjimku zachytíme pomocí bloku Try-Catch.

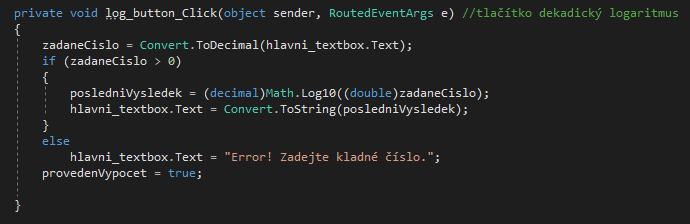
Obrázek 17 Tlačítko pro faktoriál s využitím bloků Try-Catch

Obsah obrázku interiér, snímek obrazovky

Popis vygenerován s vysokou mírou spolehlivosti

Obrázek 18 Metoda pro výpočet faktoriálu pomocí rekurze

#### Dekadický logaritmus

Vytvoříme tlačítko pro dekadický logaritmus a ošetříme, aby vstup byl větší než 0. Pomocí metody **Math.Log**, který máme k dispozici v .NETu, vypočítáme logaritmus daného čísla. Tyto hotové matematické metody jsou datového typu Double, je tedy nutné naše proměnné přetypovat.

Obrázek 19 Tlačítko pro logaritmus s ošetřením

#### Mocnina

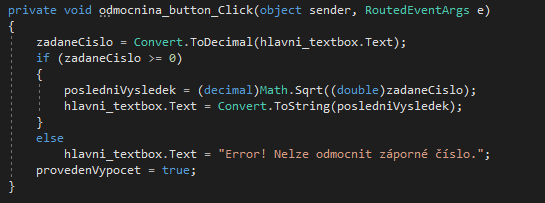
Než abychom si vytvořili tlačítko pro druhou mocninu, vytvoříme si rovnou tlačítko pro y mocninu a její metodu. Využijeme hotovou metodu **Math.Pow**. Opět budeme potřebovat proměnnou posledního zadaného čísla, zadaného čísla a posledního výsledku (stejně jako uzákladních operací, protože využijeme operaci mocniny tak, že nejprve zadáme základ, stiskneme tlačítko a zadáme mocninu. Pro tuhle operaci si přiřadíme klávesu Y.Obsah obrázku snímek obrazovky

Popis vygenerován s velmi vysokou mírou spolehlivosti

Obrázek 20 Metoda pro mocninu

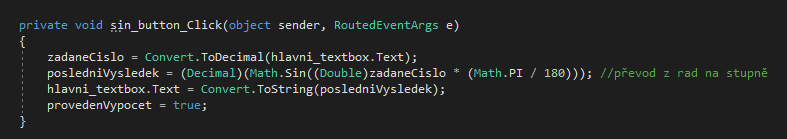
Klávesu Y přidáme i do metody **ZjistiKlavesu()** i do metody **Vypocitej().**

#### Druhá odmocnina

Postup pro vytvoření tlačítka druhé odmocniny bude podobný jako u předchozích tlačítek. Využijeme metodu **Math.Sqrt** a ošetříme zadávání záporných čísel.

Obrázek 21 Tlačítko pro druhou odmocninu

#### Goniometrické funkce

Pro goniometrické funkce využijeme hotové metody z frameworku .NET: **Math.Sin, Math.Cos, Math.Tan**. Po výpočtu se nám výsledek zobrazí v radiánech, my si to však můžeme převést rovnou na stupně.

Obrázek 22 Příklad pro tlačítko goniometrické funkce, konkrétně pro sinus

### Ošetření podruhé

Jak už jsem se dříve zmínila, musíme pořád myslet na to, že se pohybujeme v omezeném prostoru datového typu **Decimal**. Když zadáme velké číslo a vynásobíme ho dalším velkým číslem, náš program spadne. Stejně takový případ nastane, když umocníme číslo hodně velkým číslem, nebo když odečítáme, přičítáme atd. Jednu operaci už kompletně ošetřenou máme – a to je faktoriál, kde jsme použili blok **Try-Catch**.

Tento blok tedy použijeme do všech operací, kde může dojít k **přetečení**.

Pokud se vrátíme zpátky k faktoriálu, kde využíváme rekurzi, ošetříme i nadměrně velká čísla. Když zadáme např. číslo 500 a chceme vypočítat jeho faktoriál, objeví se námi nastavený text, že faktoriál tohoto čísla by byl příliš vysoký. Když ale vepíšeme např. číslo 50000, program nám spadne. Je to tím, že 50000× metoda volá sama sebe a počítač už toho není schopný. Proto omezíme zadané číslo na 28 (náš program vyšší faktoriál nespočítá).

## Grafická úprava WPF aplikace

Ke grafické úpravě aplikace nám poslouží grafický editor a vlastnosti prvků. Můžeme si nastavit barvu pozadí kalkulačky, tlačítek, upravit text a mnoho další. Část designu jsme už udělali na začátku, když jsme si vytvořili tabulku pro tlačítka.

Aby se kalkulačka nemohla zvětšovat ani zmenšovat, nastavíme **ResizeMode** u okna na „**NoResize**“. Kdybychom chtěli okno měnit, museli bychom nastavit u všech prvků včetně okna a gridu jejich vlastnost ResizeMode na auto.

Pro úsporu místa lze vypnout rámeček okna. Tato možnost se změní nastavením vlastností **WindowStyle** na **none**. Zbavíme se tím ale veškeré manipulace s oknem, takže i tedy vypnutí. Pro vyřešení tohoto problému si můžeme přidat další tlačítko do kalkulačky pro ukončení aplikace.

Barvy tlačítek, gridu i celého okna nastavíme pomocí editoru barev.

Můžeme si změnit ikonku programu. Místo automaticky nastavené ikonky nastavíme vlastní. Doporučuji rozměry 32x32, může být i 16x16. Formát ikonky musí být .**ico**. Stiskneme „Projekt“ v horním panelu a následně „Vlastnosti VědeckáKalkulačka“. V prostředí **Ikona a manifest** si vybereme vlastní ikonku.

# Závěr

V této seminární práci jsem detailně popsala postup pro vytvoření vědecké kalkulačky i s důkladným ošetřením případných chyb. Do kalkulačky by se mohlo přidat více funkcí a operací, které by kalkulačku dále zdokonalovaly. Zároveň jsme si popsali framework Windows Presentation Foundation a jazyk XAML a naučili jsme se s nimi pracovat.

# Seznam obrázků

[Obrázek 1 Vytváření nového projektu ve Visual Studiu 14](#_Toc510205442)

[Obrázek 2 Rozhrání aplikace ve Visual Studiu 14](#_Toc510205443)

[Obrázek 3 Rozdělení okna a přidání objektů 17](#_Toc510205444)

[Obrázek 4 Deklarace globálních proměnných 18](#_Toc510205445)

[Obrázek 5 Kód pro Input 18](#_Toc510205446)

[Obrázek 6 Přiřazení čísel k tlačítkům 19](#_Toc510205447)

[Obrázek 7 Metoda pro základní operace 19](#_Toc510205448)

[Obrázek 8 Příklad metody pro operaci sčítání 20](#_Toc510205449)

[Obrázek 9 Metoda pro výpočet 20](#_Toc510205450)

[Obrázek 10 Metoda pro tlačítko C 21](#_Toc510205451)

[Obrázek 11 Zákaz psaní do Textboxu v okně 21](#_Toc510205452)

[Obrázek 12 Metoda pro zjištění klávesy 22](#_Toc510205453)

[Obrázek 13 Metoda pro zjištění klávesy na numerické klávesnici 22](#_Toc510205454)

[Obrázek 14 Tlačítko pro desetinnou čárku 23](#_Toc510205455)

[Obrázek 15 Přiřazení klávesy Enter pro výpočet 23](#_Toc510205456)

[Obrázek 16 Tlačítko backspace se znakem Erase to the left 24](#_Toc510205457)

[Obrázek 17 Tlačítko pro faktoriál s využitím bloků Try-Catch 24](#_Toc510205458)

[Obrázek 18 Metoda pro výpočet faktoriálu pomocí rekurze 25](#_Toc510205459)

[Obrázek 19 Tlačítko pro logaritmus s ošetřením 25](#_Toc510205460)

[Obrázek 20 Metoda pro mocninu 26](#_Toc510205461)

[Obrázek 21 Tlačítko pro druhou odmocninu 26](#_Toc510205462)

[Obrázek 22 Příklad pro tlačítko goniometrické funkce, konkrétně pro sinus 27](#_Toc510205463)

# Reference

1. **Virius, Miroslav.** *C# 2010 - Hotová řešení.* Brno : Computer Press, 2012. ISBN 978-80-251-3730-7.

2. **Čápka, David.** itnetwork.cz. *1. díl - Úvod do WPF (Windows Presentation Foundation).* [Online] 4. Prosinec 2013. [Citace: 5. Prosinec 2017.] https://www.itnetwork.cz/csharp/formulare/wpf/c-sharp-tutorial-wpf-uvod-a-prvni-formularova-aplikace.

3. **Neznámý.** Unicode Character 'ERASE TO THE LEFT' (U+232B). *fileformat.info.* [Online] [Citace: 1. 1 2018.] https://www.fileformat.info/info/unicode/char/232b/index.htm.

4. —. Faktoriál. *cs.wikipedia.org.* [Online] [Citace: 2. 1 2018.] https://cs.wikipedia.org/wiki/Faktori%C3%A1l.