Pavel Čáp

Programování Windows 8 aplikací

pro začátečníky

Microsoft

Obsah

Poděkování

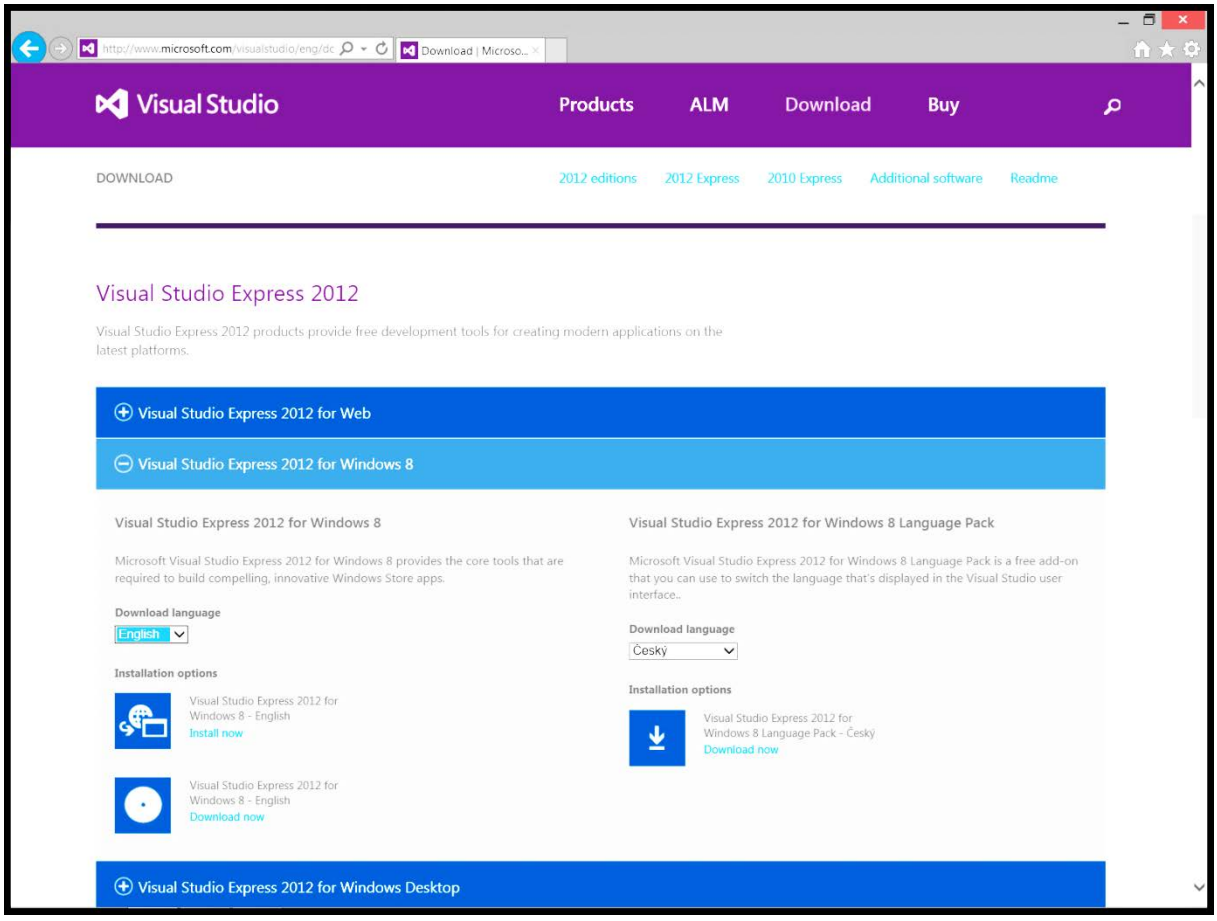
Rád bych touto cestou poděkoval Pavlu Ježkovi a Matěji Zmítkovi za pomoc při tvorbě této knížky a samozřejmě také svojí rodině, bez jejíž pomoci a trpělivosti by tato knížka nevznikla.

Úvod a instalace

Tato knížka je určena všem, kteří rádi poznávají nové věci. Je psána tak, aby i člověk, jenž nikdy neprogramoval, dokázal vytvořit zajímavé programy. Pokud máte v oblibě stavebnice a rádi poznáváte, co která součástka znamená a kam patří, tak věřte, že programování je jedna velká stavebnice s neuvěřitelným množstvím dílů. Postupně budete objevovat nové a nové věci a budete mít radost z každého nového programu. Vždy jsem se moc těšil na první rozbalení krabice, a tak nebudu zdržovat a pustíme se do toho.

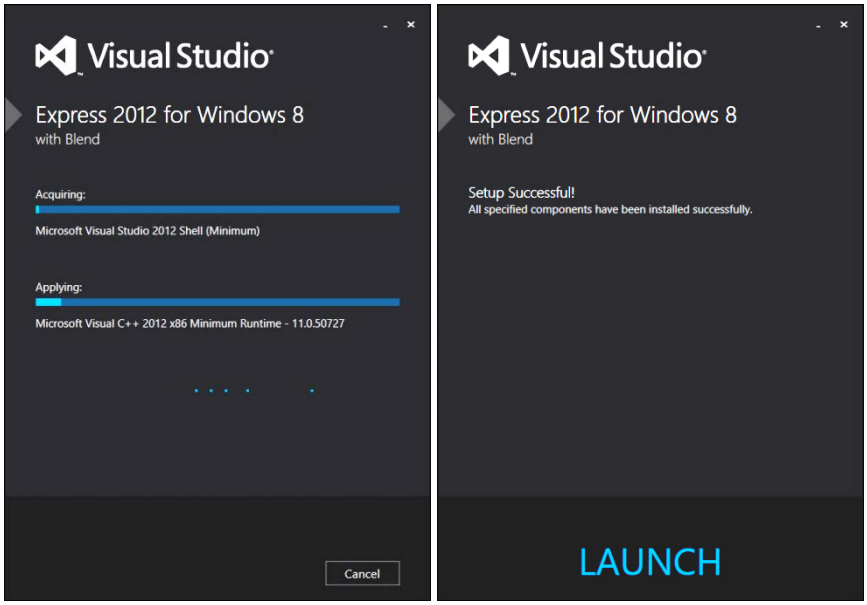
Co budeme pro naši práci potřebovat? Nejdříve je nutné zjistit, jestli máte nainstalován operační systém Windows 8. Poznáte jej podle toho, že jako jediný – na rozdíl od předešlých verzí – po startu systému neukáže pracovní plochu. Místo toho zobrazí dlaždice s aplikacemi.

Další důležitou věcí je instalace Visual Studia Express 2012 od firmy Microsoft, které je skvělé a v express edici úplně zdarma. Najdete ho na stránkách Microsoftu nebo ho můžete rovnou stáhnout z tohoto odkazu: http://www.microsoft.com/visualstudio/eng/downloads#d-express-windows-8.



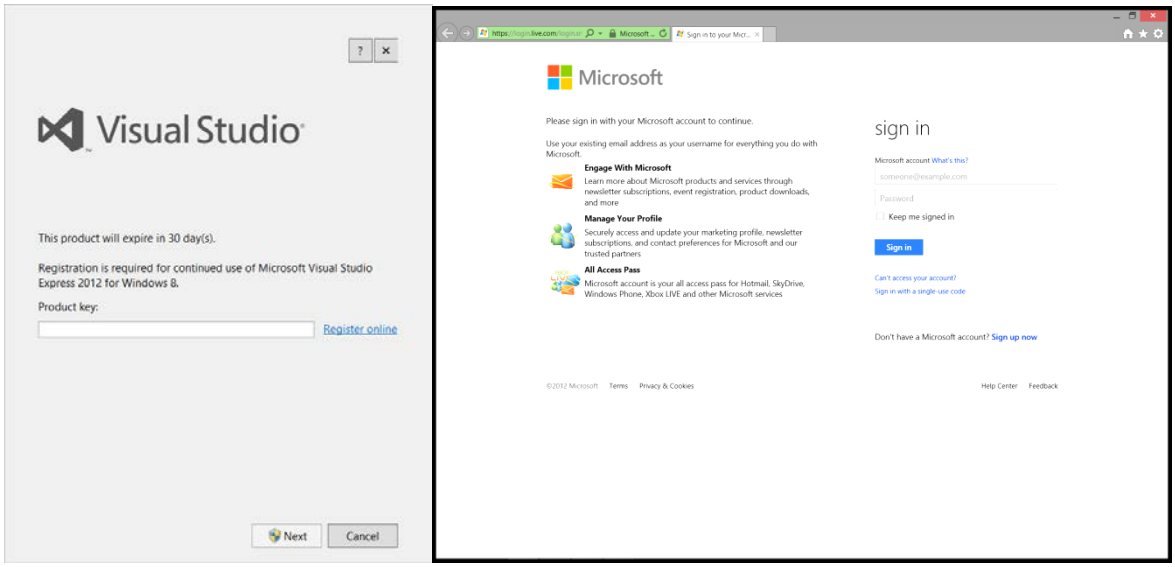
Webová stránka pro stažení Visual Studia 2012

Nejprve je třeba stáhnout a spustit instalační balíček. Postupně proběhne stahování jednotlivých částí Visual Studia, viz obrázek níže. O stavu instalace budete průběžně informováni. Nakonec je třeba vybrat jako základní jazyk C#, ve kterém budeme psát naše budoucí programy.



Průběh instalace

Pak už se zbývá jen zaregistrovat, zadat produktový klíč a můžeme začít programovat. Klikněte na odkaz Register online, a pokud ještě nemáte vytvořen Live Id účet, pak si ho vytvořte zde: https://login.live.com. Tento účet se vám bude hodit i pro jiné programy a služby od firmy Microsoft.

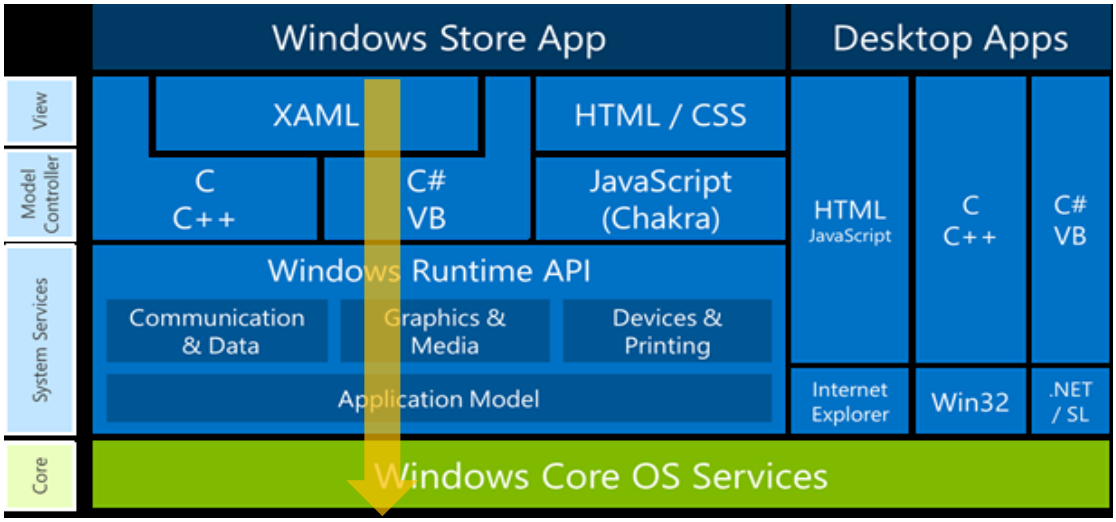


Registrace a vytvoření účtu

Doufám, že se vše podařilo, a v další kapitole se už pustíme do průzkumu programovacího prostředí.

Výběr programovacího jazyka

Programy, které budeme vytvářet, budou pracovat v novém běhovém prostředí. Toto běhové prostředí bylo vytvořeno pro aplikace s označením Windows Store App. Jednotlivé aplikace se instalují z centrálního uložiště (obchodu). V základu si můžete vybrat jeden ze čtyř programovacích jazyků. Pro naši práci si zvolíme programovací jazyk C# v kombinaci s XAML. C#, nebo také Csharp, vznikl již v roce 2002, vychází svojí syntaxí z jazyků C++ a Java. Jedná se moderní vysokoúrovňový objektový jazyk. XAML je značkovací jazyk používaný k vytváření grafického rozhraní aplikace. Proč dva jazyky? Nikdy není na škodu oddělit běhovou část programu od grafického rozhraní. Málokterý programátor dokáže vytvářet pěknou grafiku a málokterý grafik se orientuje v běhovém kódu. Microsoft tyto dvě části oddělil a grafikům dal program Blend, který dokáže generovat XAML do vašeho programu. Jak spustit Blend si povíme později. Pro představu o vnitřní struktuře běhového prostředí se podívejte na následující obrázek.



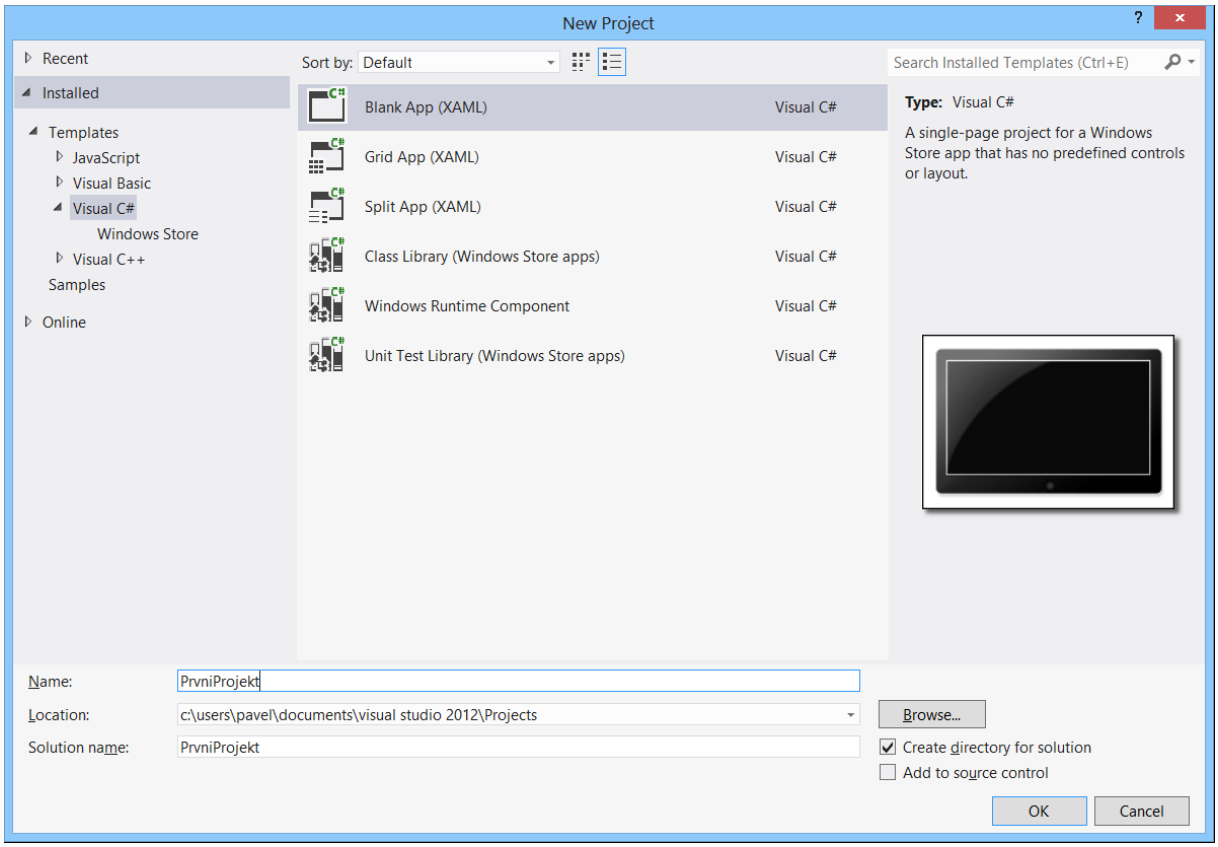
Šipka na obrázku ukazuje námi zvolenou variantu – pěkně odshora

1. Vytváříme program pro Windows Store App.
2. Pro zobrazení použijeme XAML.
3. Běh programu napíšeme v jazyce C#.
4. V systému musí být Windows Runtime API pro Windows Store App.
5. Vše běží na jádru Windows 8.

Pokud jste nepřeskočili předchozí kapitolu, máte vše zmíněné nainstalováno a můžeme začít.

Prohlídka programovacího prostředí

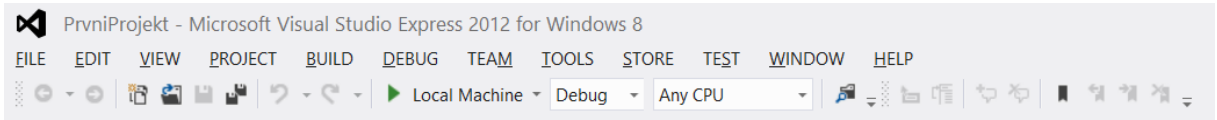
Nastal čas vytvořit první projekt. Zapneme Visual Studio 2012 Express. V menu vybereme *File → New Project*. Zvolíme jazyk *Visual C# → Blank App (XAML).* V řádku *Name* vyplníme název projektu *PrvniProjekt* a zkontrolujeme *Location* cestu, kde bude projekt uložen. Nakonec potvrdíme *OK*.



Založení nového projektu

Dále si popíšeme jednotlivé části rozložení obrazovky našeho projektu.

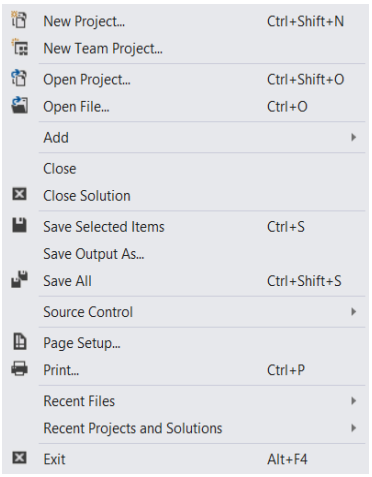
Menu – základní prostředek pro výběr jednotlivých nabídek. Nebudeme zde popisovat dopodrobna všechny nabídky, vybereme jen ty, které budeme v této chvíli potřebovat.



Základní lišta menu

File

Menu → *File* – slouží pro práci s projekty i s jednotlivými soubory.



→ *New Project…* – vytvoření nového projektu

→ *Save All* – uložení všech souborů projektu

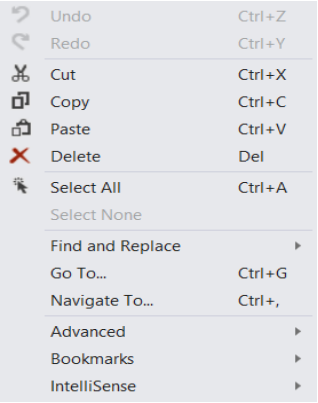
→ *Recent Files* – otevření posledního souboru

→ *Recent Projects …* – otevření posledního projektu

→ *Exit* – ukončení Visual Studia

Edit

Menu → *Edit* – slouží především pro editaci a vyhledávání v textu programu.



→ *Cut* – vyjmout označenou část

→ *Copy* – kopírovat označenou část

→ *Paste* – vložit označenou část

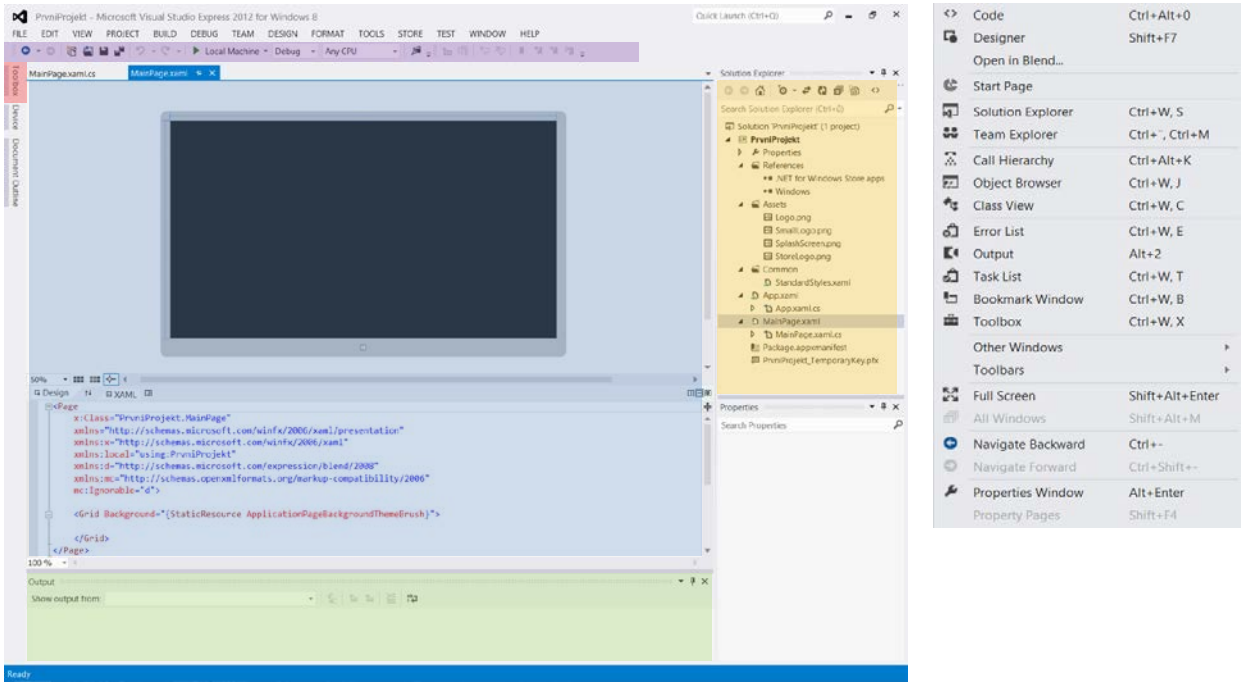
→ *Delete* – smazat označenou část

→ *Select All* – označit vše

→ *Find and Replace* – vyhledat nebo nahradit text

View

Menu → *View* – slouží k zapnutí nebo vypnutí jednotlivých pracovních panelů na obrazovce nebo k přepínání jednotlivých typů souborů.



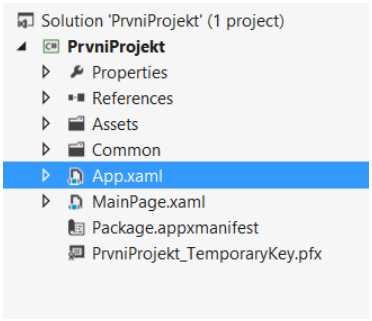
Přehled zobrazených panelů ve vývojovém prostředí a menu View

View → *Code* – přepne na kód aplikace, zobrazuje soubory s koncovkou .cs

→ *Designer* – přepne na návrhové zobrazení, koncovka .xaml

→ *Open in Blend* – otevře projekt v Blendu pro vytváření grafického rozhraní

View → *Solution Explorer*

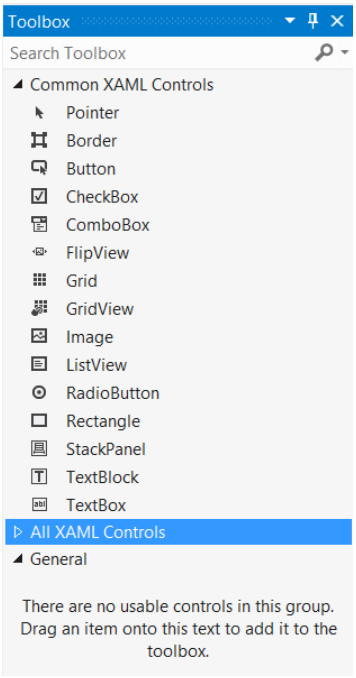


Solution Explorer – slouží k přehledu a správě souborů a složek v našem projektu. Přehledně zobrazuje všechny soubory včetně referencí a obrázků. Standardně najdete projekt ve složce Dokumenty/visual studio 2012/Projects/…. Tato cesta se zadává při vytváření nového projektu.

View → *Properties*

Properties – jedná se o panel pro zobrazování vlastností a událostí jednotlivých komponent, které naleznete v Toolboxu. Vlastnosti jsou věci, jež popisují například vzhled, umístění a velikost komponenty. Události se vážou k chování komponenty, popis a vysvětlení událostí budeme řešit později.

View → *Toolbox*



Toolbox – slouží k výběru XAML komponent, které můžete použít ve svých programech. Komponentu jednoduše přetáhnete myší nebo provedete dvojklik. Později si popíšeme a ukážeme nejpoužívanější z nich.

View → *Output*

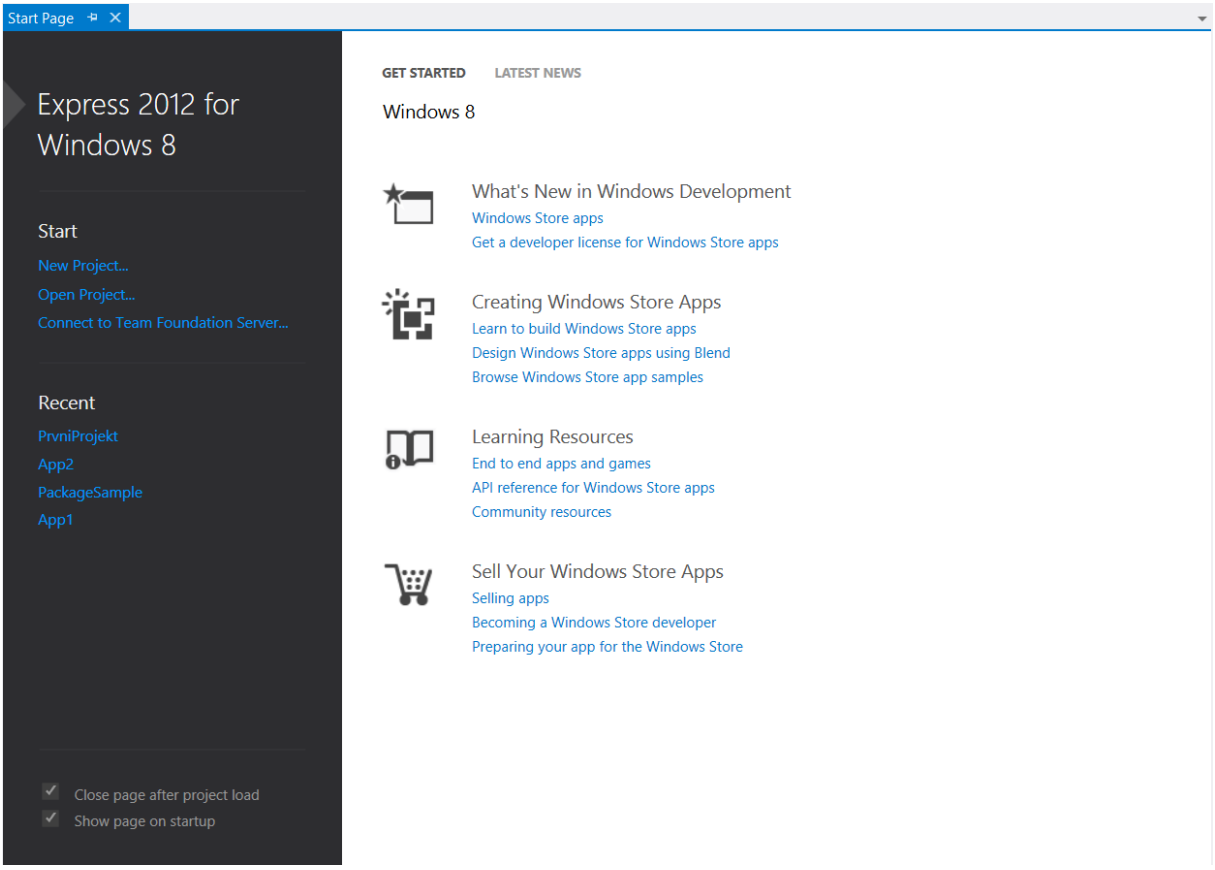
Output – výstupní panel, který informuje o spouštění, překladu, varováních a chybách aplikace. Je to užitečný pomocník při odlaďování programu.

View → *Toolbars*

Toolbars – nástrojová lišta, kterou lze upravovat podle vašich požadavků. Můžete přidávat, ubírat nebo přemisťovat jednotlivé nástroje tak, aby se vám ve Visual Studiu 2012 pracovalo co nejlépe.

View → *Start Page*

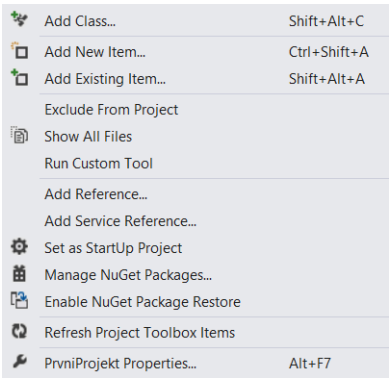
Start Page – přepne na úvodní obrazovku, ze které můžete stejně jako z menu založit nový projekt, otevřít jiný projekt nebo se podívat na minulé projekty. Navíc máte možnost se podívat na předpřipravené projekty, učit se a dozvědět se více o Windows Store.



Obrázek Start Page

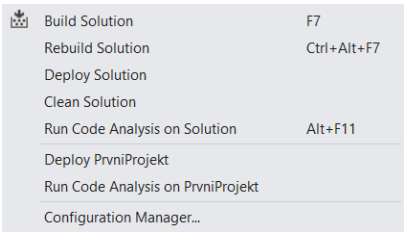
Project

Menu → Project – slouží k práci s projektem, k přidávání a úpravě souborů projektu. Je možné také nastavovat a upravovat vlastnosti projektu. Všechny položky lze nastavovat přes *Solution Explorer*.



Build

Menu → Build – toto menu slouží k sestavení a kontrole vašeho programu. Při psaní kódu ve Visual Studiu dochází k automatickému vyhledávání syntaktických chyb (chyby ve špatném napsání příkazů) již během psaní. Výstup je automaticky směrován do panelu *Output*.

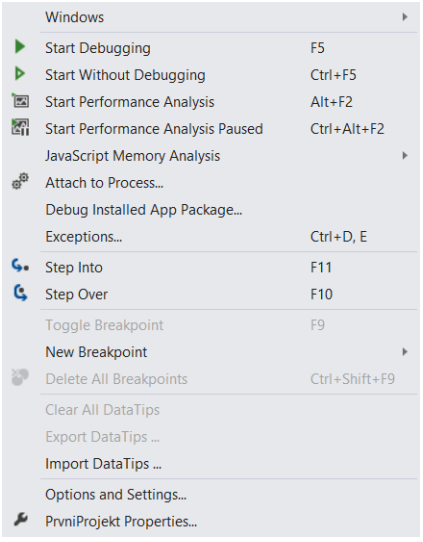


Výstup může vypadat třeba takto:

```
1>------ Deploy started: Project: PrvniProjekt, Configuration: Debug Any CPU ------
1>Updating the layout...
1>Copying files: Total <1 mb to layout...
1>Deployment complete. Full package name: "4dfabd6b-7115-4a7a-97d5-
dbc7c2faa55d_1.0.0.0_neutral__208q3t7qnpxrj"
========== Deploy: 1 succeeded, 0 failed, 0 skipped ==========
```

Debug

Menu → Debug – toto menu slouží k puštění a odladění programu. Je zde možné nastavovat tzv. Breakpointy, které umožňují zastavení programu a zjištění například hodnot proměnných. Při spuštění programu dochází nejprve ke kontrole a sestavení a posléze i k vytvoření spustitelného souboru.



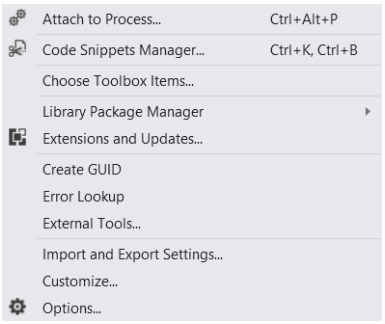
Debug → Start Debugging – spuštění s možností ladění programu.

Debug → Start Without Debugging – spuštění bez možnosti ladění programu.

Debug → Step Into – možnost postupného krokování programu po jednotlivých příkazech.

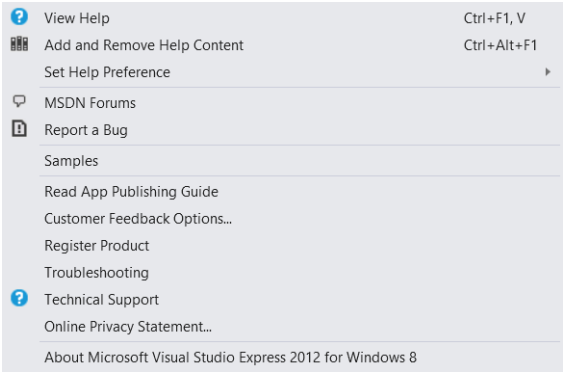
Tools

Menu → Tools → Option – velmi detailní nastavení programovacího prostředí. Tip – zkuste si pro začátek nastavit *Color Theme* z *Dark* na *Light*.



Help

Menu → Help – zde najdete nápovědu a pomoc při programování.



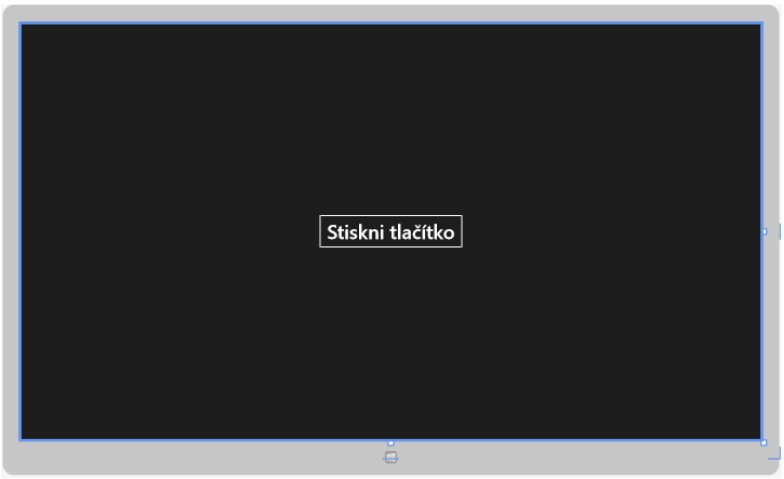
Pozorný čtenář si určitě všiml, že jsme přeskočili několik položek menu. Tato menu jsou stejně důležitá jako menu, která jsme prošli, ale pro vytvoření našeho prvního projektu je nebudeme potřebovat.

První projekt

Pokud jste si pečlivě prohlédli prostředí Visual Studia, pustíme se do dokončení naší aplikace. Většina knih o programování začíná aplikací „Ahoj Světe“ a my nebudeme výjimkou. Jen to trochu vylepšíme o jedno tlačítko.

1. V Solution Exploreru klikněte dvakrát na soubor MainPage.xaml.
2. Po levé straně najděte záložku ToolBox a vyhledejte komponentu Buttton (Tlačítko).
3. Na Button dvakrát klikněte nebo jej myší přetáhněte na návrhové zobrazení.
4. Po levé straně najděte záložku ToolBox a vyhledejte komponentu TextBlock.
5. Na TextBlock dvakrát klikněte nebo jej myší přetáhněte na návrhové zobrazení.
6. Upravte XAML takto:

```
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">

<Button Content="Stiskni tlačítko"
 HorizontalAlignment="Center"
 VerticalAlignment="Center"
 FontSize="36"
 Click="Button_Click_1"  />


<TextBlock HorizontalAlignment="Center"
 VerticalAlignment="Center"
 Name="Textik"
 FontSize="36"
 Margin="0,160,0,0"
 TextWrapping="Wrap" />
</Grid>
```



Návrhové zobrazení

7. Dvakrát klikněte na tlačítko „Stiskni tlačítko“ a otevře se záložka MainPage.xaml.cs.

8. Zde do metody Button_Click_1, která se sama vygenerovala, napište následující text. Tato metoda se zavolá vždy, když zmáčknete tlačítko:

```
private void Button_Click_1(object sender, RoutedEventArgs e)
        {
            Textik.Text = "Ahoj Světe";
        }
```

9. Upravili jsme vlastnost komponenty TextBlock, kterou jsme nazvali v XAMLu Textik. Do vlastnosti Text jsme přiřadili řetězec „Ahoj Světe“.
10. Tak, a máme hotovo, stačí stisknout Ctrl+F5 nebo zvolit menu *Debug → Start Without Debugging*.
11. Náš program běží, po stisku tlačítka se zobrazí „Ahoj Světe“.

Program „Ahoj Světe“

Pokud se vám vše povedlo, tak jste se právě stali programátory. Máte za sebou první opravdový program. Jistě ještě úplně všemu nerozumíte, ale toho se nebojte, v dalších kapitolách si všechno vysvětlíme.

Grafické rozhraní

Jazyk Xaml se v našich aplikacích používá pro tvorbu grafického rozhraní. Jeho výhodou je jednoduchost a přehlednost. V tomto dílu knihy se nebudeme zajímat o možnosti nastavení a úprav grafického rozhraní aplikace, spíše se zaměříme na tvorbu programu. Pokud nevíte, jak napsat základní kód programu, nepomůže vám barevný vzhled aplikace. Přesto si popíšeme základy, které budeme pro naše programy potřebovat. Vzhledem k tomu, že se dnes programy z Windows Store instalují na různých zařízeních (počítače, tablety), bylo nutné definovat základní pohledy, ve kterých naše aplikace poběží.

Landscape – *aplikace přes celou obrazovku na šířku*

Portrait – *aplikace přes celou obrazovku na výšku*

Snapped – *aplikace je připnutá k okraji*

Filled – *aplikace jako zbytek po připnutí*

Kvůli těmto různým typům zobrazení není vhodné umisťovat grafické komponenty na pevné pozice. Raději je umístíme do Gridu a zarovnáme na určené místo.

Grid

Základním grafickým typem je komponenta Grid. Jedná se o mřížku, do které se umísťují další grafické prvky. Dokáže rozdělit obrazovku na oddělené části s různými vlastnostmi.

```
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">

        <Grid.RowDefinitions>
            <!-- řádek 0 "*" maximální výška -->
            <RowDefinition Height="*"/>
            <!--  řádek 1 "Auto" výška se automaticky přizpůsobí komponentám -->
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <Grid.ColumnDefinitions>
            <!--  "*2" "*1" sloupce budou ve velikostním poměru 2:1 -->
            <ColumnDefinition Width="2*"/>
            <ColumnDefinition Width="1*"/>
     <!--  sloupec s pevně danou velikostí 100px -->
     <ColumnDefinition Width="100"/>
        </Grid.ColumnDefinitions>

<!--  tlačítko bude umístěno ve druhém sloupci (Grid.Row="1") a druhém řádku
(Grid.Column="1") -->
        <Button FontSize="34" Content="Tlačítko" HorizontalAlignment="Left"
  Margin="0,0,0,0" VerticalAlignment="Top" Grid.Row="1" Grid.Column="1"
  Height="100" Width="256"/>

</Grid>
```

Příklad rozdělení základního Gridu na čtyři části. Řádky i sloupce se indexují od nuly.

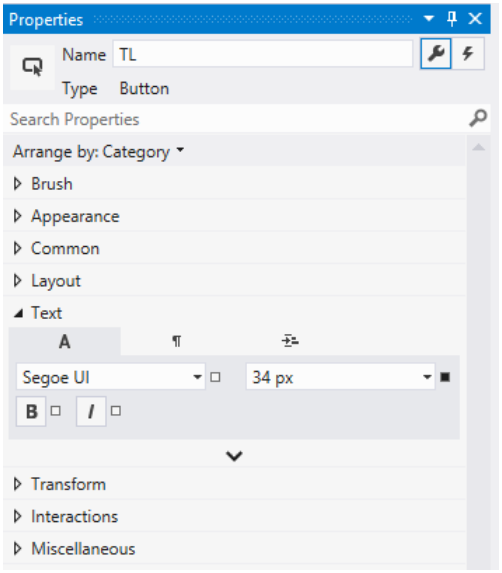
Button

Tlačítko je jednou ze základních komponent, kterou budeme v našich programech potřebovat. Ze základních vlastností vybereme následující:

```
<Button

 Name="TL"
 Content="Stiskni tlačítko"
 HorizontalAlignment="Center"
 VerticalAlignment="Center"
 Margin="0,0,0,0"
 FontSize="36"
 Height="100"
 Width="256"

 Click="Button_Click_1"  />
```

Vlastnost	Popis
Name	Jméno tlačítka
Content	Popis tlačítka
HorizontalAlignment	Horizontální zarovnání
VerticalAlignment	Vertikální zarovnání
Margin	Zarovnání v rámci vnitřního prvku
Height	Šířka tlačítka
Width	Výška tlačítka

Nesmíme zapomenout na událost `Click`, která po stisku tlačítka zavolá metodu s názvem `Button_Click_1`.



Vlastnosti tlačítka na panelu Properties

TextBlock

Tato komponenta slouží k výpisu textu na obrazovku. Z jejích vlastností je asi nejdůležitější Name (její jméno) a Text, kam zadáváme text ve formě řetězce znaků (string). Vlastnost FontSize nastavuje velikost písma. Každá komponenta obsahuje velké množství dalších vlastností, na které se můžete podívat v panelu Properties. Samozřejmě je možné se podívat i na události: stačí v panelu Properties vybrat místo symbolu klíče blesk. Nezapomeňte nejprve kliknout na komponentu v návrhovém zobrazení, u níž vás dané vlastnosti zajímají.

```
<TextBlock  Name="Vysledek" HorizontalAlignment="Center"
 VerticalAlignment="Center" FontSize="34" Margin="0,160,0,0"
 Text=" " />
```

TextBox

Tato komponenta se používá pro zadávání textu. Zadaný text ve formě řetězce znaků je možné vyčíst z vlastnosti Text a maximální délku zadávaného textu lze nastavit pomocí vlastnosti MaxLength.

```
<TextBox Name="Procenta" HorizontalAlignment="Center"
 VerticalAlignment="Center" Margin="150,-150,0,0" FontSize="36"
 MaxLength="4" Text="0"  />
```

Ve Windows Store aplikacích můžete používat celou řadu dalších komponent, které najdete na panelu ToolBox. Pro naše programy si zatím vystačíme s výše popisovanými, ale postupem času si určitě oblíbíte i další.

Základní datové typy

Každý program, který napíšete, bude obsahovat nějakou proměnou. Proměnná není nic jiného než pojmenované místo v operační paměti počítače. Paměť počítače si můžete představit jako hodně velikou tabulku, kde v každé buňce může být uložena pouze 1 nebo 0 – třeba jako v následující tabulce:

1	1	0	1	0	1	1	0	0	0	1	0	1	1	0	1	0	1	0	1	1	1	1	1	0	0	0	1	0	1	0	1
1	0	1	0	1	1	1	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1	1
1	0	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Jedné buňce se říká bit, a pokud spojíme dohromady osm bitů, dostaneme jeden byte. Každá hodnota uložená v paměti musí být poskládána jenom z jedniček a nul. Jak ale zapíšeme třeba číslo 10? Je to docela jednoduché. Musíte si představit, že každá buňka má také přiřazenu nějakou hodnotu. Hodnota bitu je odvozena od binární soustavy $2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, …

Hodnota	128	64	32	16	8	4	2	1
Bit	0	0	0	0	1	0	1	0

Stačí sečíst hodnoty bitů, kde jsou jedničky, a dostaneme číslo v desítkové soustavě (soustava ve které počítáme), v našem případě 8 + 2 = 10. Potom je desítka v paměti zapsána jako 1010.

Jak bude vypadat číslo 7?

Hodnota	128	64	32	16	8	4	2	1
Bit	0	0	0	0	0	1	1	1

Jak bude vypadat číslo 65?

Hodnota	128	64	32	16	8	4	2	1
Bit	0	1	0	0	0	0	0	1

Možná jste si všimli, že pokud číslo končí jedničkou, tak je liché. Naopak s 0 je sudé. Jaké největší číslo se vejde do jednoho bytu (osmi bitů)? Musíme dát všude jedničky, a pokud budete dobře počítat, vyjde vám 255.

Samozřejmě to jde spočítat i jednodušeji: $2^n - 1$, kde n je počet bitů:

$2^8 - 1 = 256 - 1 = 255$

V případě čísel větších než 255 stačí přidat další bity a můžeme uložit libovolná čísla. Například číslo 577 potom bude 1001000001:

Hodnota	512	256	128	64	32	16	8	4	2	1
Bit	1	0	0	1	0	0	0	0	0	1

Každá proměnná v počítači je nějakého datového typu. Datové typy určují, co do proměnné můžeme uložit, a na základě toho alokují (zaberou) potřebnou velikost paměti. Základní datové typy si teď ukážeme.

Celočíselné datové typy

Integer

Do celočíselného datového typu se ukládají celá čísla. Nejpoužívanější datový typ je Integer, jehož velikost činí v základním tvaru 4 byty (32 bitů). Použití si ukážeme na jednoduchém programu. Použijeme naši aplikaci PrvniProjekt:

```
int A = 10;
```

Tento příkaz založí celočíselnou datovou proměnou typu Integer s názvem A a uloží do proměnné hodnotu 10. Znaménko „=“ neznamená rovnost, ale „zapiš hodnotu z pravé strany (10) do proměnné s názvem A“. Do této proměnné můžete uložit čísla kladná i záporná do velikosti $2^{31} - 1$ (−2 147 483 648 až 2 147 483 647). Jeden bit je použit pro znaménko, proto jenom 2^{31}.

Nyní upravíme projekt:

```
private void Button_Click_1(object sender, RoutedEventArgs e)
        {
            int A = 10;
            Textik.Text = A.ToString();
        }
```

Po stisku tlačítka se místo „Ahoj Světe“ vypíše hodnota proměnné A:

Byte

Jedná se také o celočíselný datový typ. Do bytu lze uložit celá čísla v rozmezí 0–255. Po úpravě bude náš program vypadat následovně:

```
private void Button_Click_1(object sender, RoutedEventArgs e)
        {
            byte A = 10;
            Textik.Text = A.ToString();
        }
```

Jeho funkce bude úplně stejná, ale ušetříme místo v paměti.

Long

Pokud budeme potřebovat uložit velmi velké celé číslo, můžeme použít datový typ Long. Jeho velikost je 8 bytu (64 bitů). U většiny programů si vystačíme s Integerem, ale přesto se Long někdy může hodit. Do této proměnné můžeme uložit čísla kladná i záporná do velikosti $2^{63} - 1$ (−9 223 372 036 854 775 808 až 9 223 372 036 854 775 807).

```
private void Button_Click_1(object sender, RoutedEventArgs e)
        {
            long A = 10;
            Textik.Text = A.ToString();
        }
```

Program „Počítadlo“

Zkusme teď udělat program, který po stisku tlačítka připočte jedničku do celočíselné datové proměnné a hodnotu zobrazí.

1. Založte nový projekt s názvem Pocitadlo.
2. Pokud si ještě nepamatujete, jak umístit komponentu tlačítka a textbloku, podívejte se na předešlý projekt a proveďte kroky 1–6.
3. U šestého kroku změňte vlastnost `Content="Připočti"`.
4. Dvakrát klikněte na Tlačítko „Připočti“ a otevře se záložka MainPage.xaml.cs.
5. Přidejte do třídy MainPage proměnnou typu Integer a uložte do ní hodnotu 0. Proměnné, které jsou uvnitř třídy, se nazývají datové položky.
6. Následně upravte metodu Button_Click_1, která se zavolá po stisku tlačítka:

```
int pocet = 0;

private void Button_Click_1(object sender, RoutedEventArgs e)
{
    pocet = pocet + 1;
    Textik.Text = pocet.ToString();
}
```

7. Hotovo! Stačí stisknout Ctrl+F5 nebo zvolit menu *Debug* → *Start Without Debugging*.

Pokud se vše podařilo, máme hotový druhý program. Je potřeba ještě vysvětlit tento řádek:

```
pocet = pocet + 1;
```

Program pracuje tak, že nejprve vykoná pravou stranu, kde přičte jedničku do proměnné počet, a potom celý výsledek přesune do téže proměnné. Je také možné napsat tento zkrácený výraz, který udělá totéž:

```
pocet++;
```

Teď náš program ještě vylepšíme o tlačítko Reset, které vynuluje naše počítadlo.

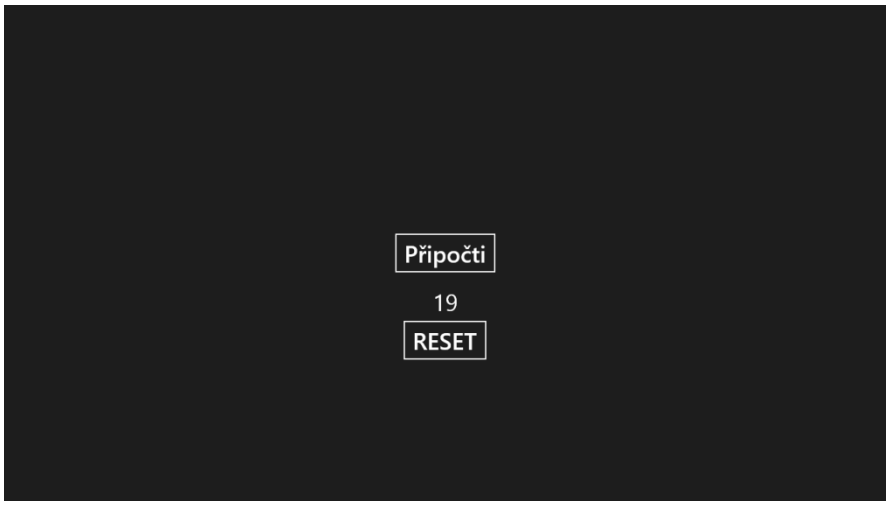
1. Upravte XAML na MainPage.xaml:

```
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">

 <Button
 Content="Připočti"
 HorizontalAlignment="Center"
 VerticalAlignment="Center"
 FontSize="36"
 Click="Button_Click_1"  />

 <Button
 Content="RESET"
 HorizontalAlignment="Center"
 VerticalAlignment="Center"
 FontSize="36"
 Margin="0,270,0,0"
 Click="Button_Click_2"  />

 <TextBlock
  Name="Textik"
 HorizontalAlignment="Center"
 VerticalAlignment="Center"
 FontSize="36"
 Margin="0,160,0,0"
 TextWrapping="Wrap"
 Text=" " />
</Grid>
```

2. Upravte kód na MainPage.xaml.cs:

```
    // založíme proměnnou pocet datového typu int
int pocet = 0;

// metoda Button_Click_1 se zavolá při stisku tlačítka Připočti
private void Button_Click_1(object sender, RoutedEventArgs e)
        {
            pocet = pocet + 1;
            Textik.Text = pocet.ToString();
        }

// metoda Button_Click_2 se zavolá při stisku tlačítka Reset
private void Button_Click_2(object sender, RoutedEventArgs e)
        {
            pocet = 0;
            Textik.Text = pocet.ToString();
            }
```

3. Hotovo! Výsledek by měl po spuštění vypadat takto:



Reálné datové typy

Reálné datové typy slouží k ukládání čísel s desetinnou čárkou. Většina programů, které mají za úkol počítat, používá tyto datové typy kvůli přesnosti výpočtu.

Double

Jedná se o nejpoužívanější datový typ pro reálná čísla. Jeho velikost v paměti je 8 bytů (64 bitů). Přesnost doublu činí 15–16 desetinných míst:

```
double prom = 10.5;
```

Pokud budete ukládat do proměnné konstantu, nezapomeňte, že desetinná čárka se zapisuje jako tečka. Tento řádek vytvoří proměnnou s názvem prom a uloží do ní hodnotu 10,5.

Float

Tento datový typ je o polovinu menší než double. Jeho velikost činí 4 byte (32 bitů). Někdy se mu také říká singl. Jeho přesnost je 7 desetinných míst.

Pozor na ukládání konstantních hodnot. Desetinné číslo se standardně přetypovává na double (double je dvakrát větší, a proto ho nelze uložit do floatu), tudíž musíme ke konstantě doplnit f (float):

```
float prom = 10.5f;
```

Decimal

Poslední reálný datový typ je největší a nejpřesnější pro ukládání desetinných čísel. Jeho přesnost je na 28-29 desetinných míst. Používá se zřídka, jen u výpočtů s nutností velmi přesných výsledných hodnot. U konstant nezapomeňte dopsat M:

```
decimal prom = 10.5M;
```

Program „Výpočet slevy“

Pokud jste si pečlivě přečetli vše o reálných datových typech, tak nastal čas si je vyzkoušet na příkladu. Program, který vytvoříme, bude umět vypočítat slevy. Zadáme současnou cenu a procenta slevy. Program zobrazí novou cenu zboží s uplatněnou slevou. Například nové boty stojí 1 000 Kč a prodavač mi dá slevu 15 %. Program následně spočítá, že nová cena činí 850 Kč.

1. Založte nový projekt s názvem Slevomatik.
2. Stejně jako v předešlých programech umístěte komponentu tlačítka (Button) a komponentu TextBlock (pro zobrazení výsledku).
3. Přidejte novou komponentu TextBox (slouží k zadávání textu):

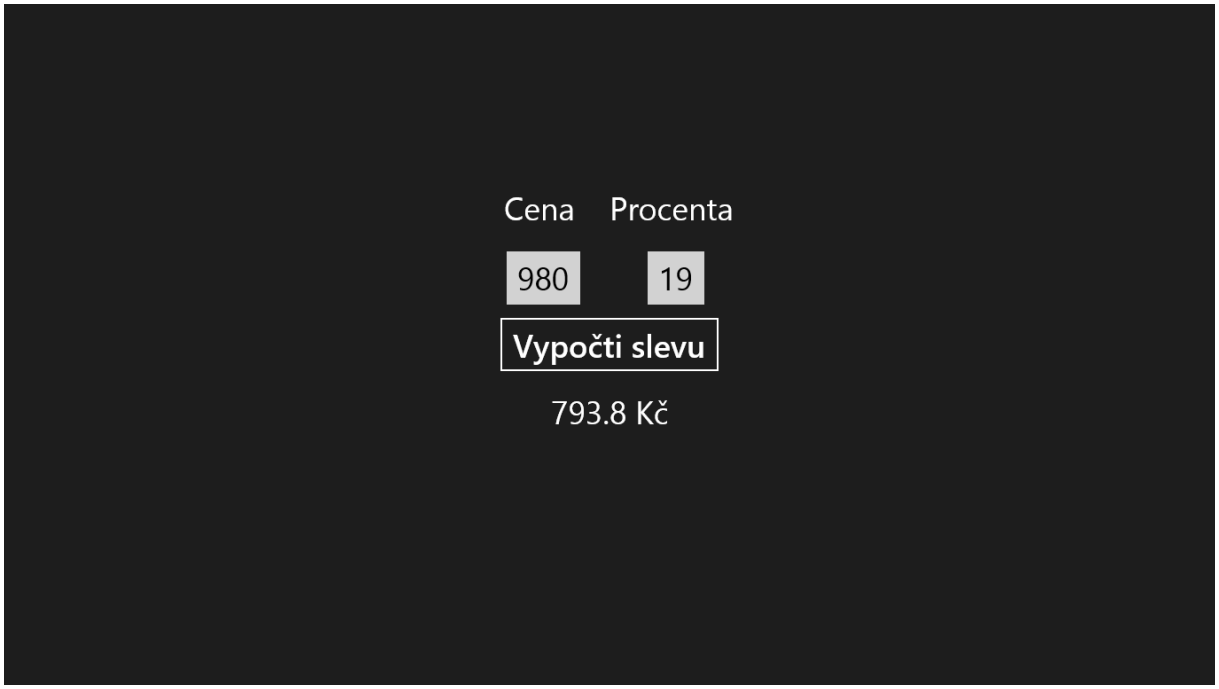
```
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">

        <Button Content="Vypočti slevu" HorizontalAlignment="Center"
  VerticalAlignment="Center" FontSize="36" Click="Button_Click_1" />

        <TextBlock  Name="Vysledek" HorizontalAlignment="Center"
  VerticalAlignment="Center" FontSize="36" Margin="0,160,0,0"
  TextWrapping="Wrap" Text=" " />

<TextBlock  HorizontalAlignment="Center" VerticalAlignment="Center"
 FontSize="36" Margin="20,-300,0,0" TextWrapping="Wrap" Text="Cena Procenta"/>

 <TextBox Name="Procenta" HorizontalAlignment="Center"
  VerticalAlignment="Center" Margin="150,-150,0,0" FontSize="36"
  TextWrapping="Wrap" Text="0" />

        <TextBox Name="Cena" HorizontalAlignment="Center" VerticalAlignment="Center"
         Margin="-150,-150,0,0" FontSize="36" TextWrapping="Wrap" Text="0" />

</Grid>
```

4. Dvakrát klikněte na Tlačítko „Vypočti slevu“ a otevře se záložka MainPage.xaml.cs.
5. Upravte program:

```
private void Button_Click_1(object sender, RoutedEventArgs e)
        {
```

```
        // založíme proměnnou cena datového typu double
        // zkonvertujeme vlastnost Text z TextBoxu s názvem Cena do double
        // a uložíme do proměnné cena

        double cena = Convert.ToDouble(Cena.Text);

        // totéž co řádek předtím, a uložíme do proměnné procenta

        double procenta = Convert.ToDouble(Procenta.Text);

        // založíme proměnnou výsledek
        // vypočítáme výsledek (/ je děleno, * je krát, - je minus) a uložíme do
        // proměnné výsledek

        double vysledek = cena/100 *(100 - procenta);

        // do TexBlocku s názvem Vysledek a jeho vlastnosti Text uložíme proměnnou
        // vysledek jako řetězec

        Vysledek.Text = vysledek.ToString() + " Kč";
    }
```

6. Hotovo! Výsledek by měl po spuštění vypadat třeba takto:



Matematické operátory

Právě jsme si prošli všechny číselné datové typy. Ještě je třeba si vysvětlit, jaké operátory u nich lze používat:

+ sčítání – sečte proměnné A a B a výsledek uloží do C:

```
int A = 13;
int B = 5;
int C = A + B;
```

- odčítání – odečte proměnnou B od A a výsledek uloží do C:

```
int A = 13;
int B = 5;
int C = A - B;
```

*** násobení** – vynásobí proměnnou A krát B a výsledek uloží do C:

```
int A = 13;
int B = 5;
int C = A * B;
```

/ dělení – vydělí A děleno B a výsledek uloží do C. Pozor u celočíselných datových typů – ořezává desetinné místo, nejedná se o zaokrouhlování 13/5 = 2!

```
double A = 13;
double B = 5;
double C = A + B;
```

% modulo – modulo je celočíselný zbytek po dělení. Příklady:

5%3 = 2 trojka se do pětky vejde jednou a zbudou 2

7%3 = 1 trojka se vejde do sedmičky dvakrát a zbude 1

```
int A = 13;
int B = 5;
int C = A % B;
```

Operátory mohou být i pro porovnání, ale ty probereme později u podmíněných příkazů.

Třída Math

Než budeme pokračovat dále, bylo by dobré si ukázat jednu velmi zajímavou třídu, která se jmenuje Math a obsahuje spoustu užitečných metod pro počítání.

Odmocnina

Pokud budete například potřebovat druhou odmocninu, použijete metodu **Sqrt()**:

```
int A = 16;
double cislo = Math.Sqrt(A);
```

V proměnné číslo bude uložena hodnota 4.

Mocnina

Pro výpočet mocniny je připravena metoda **Pow()**:

```
int A = 10;
double cislo = Math.Pow(A,2);
```

První parametr je číslo, které chceme umocnit, a druhý parametr vyjadřuje na kolikátou. V našem případě je to 10^2, což znamená, že v proměnné číslo bude uložena hodnota 100.

Absolutní hodnota

Pro výpočet absolutní hodnoty je připravena metoda **Abs()**:

```
int A = -10;
int cislo = Math.Abs(A);
```

Hodnota v proměnné číslo je 10.

Goniometrické funkce

Pro výpočet goniometrických funkcí lze použít metodu **Sin()**, **Cos()**, **Tan()** a další. Pozor, je třeba si uvědomit, že vstupní parametry se zadávají v radiánech. Pro přepočet ze stupňů na radiány můžete zvolit následující vzorec:

```
radian = (stup * Math.PI)/180
```

V uvedeném příkladu je vypočítán sin 90 stupňů a výsledná hodnota 1 je uložena v proměnné výsledek:

```
double stup = 90;
double radian = (stup * Math.PI)/180;
double vysledek = Math.Sin(radian);
```

Zaokrouhlování

Pro zaokrouhlování čísel je možné použít metodu **Round(A,B)**. Vstupní parametr A je zaokrouhlované číslo a vstupní parametr B znamená na kolik desetinných míst:

```
double A = 9.458;
double vysledek = Math.Round(A,2);
```

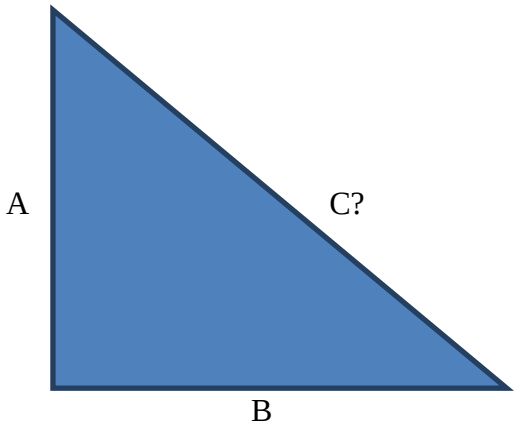
Potom ve výsledku bude uložena zaokrouhlená hodnota 9,46.

Třída Math obsahuje velké množství dalších metod, které si můžete projít a vyhledat. Úplný výpis metod s popisem naleznete na odkazu: http://msdn.microsoft.com/cs-cz/library/system.math.aspx.

Program „Pythagorova věta“

V této chvíli raději zkusíme vytvořit příklad, ve kterém použijeme některé z uvedených metod. Náš program bude počítat podle Pythagorovy věty délku přepony v pravoúhlém trojúhelníku.

Máme pravoúhlý trojúhelník tvořený odvěsnami o délce A a B. Naším úkolem je zjistit délku přepony C. Budeme vycházet ze vzorce $C^2 = A^2 + B^2$.



1. Založte nový projekt s názvem Pythagoras.
2. Stejně jako v předešlých programech umístěte komponentu tlačítka (Button) a komponentu TextBlock (pro zobrazení výsledku).
3. Přidejte komponentu TextBox (slouží k zadávání textu).

4. Upravte Xaml:

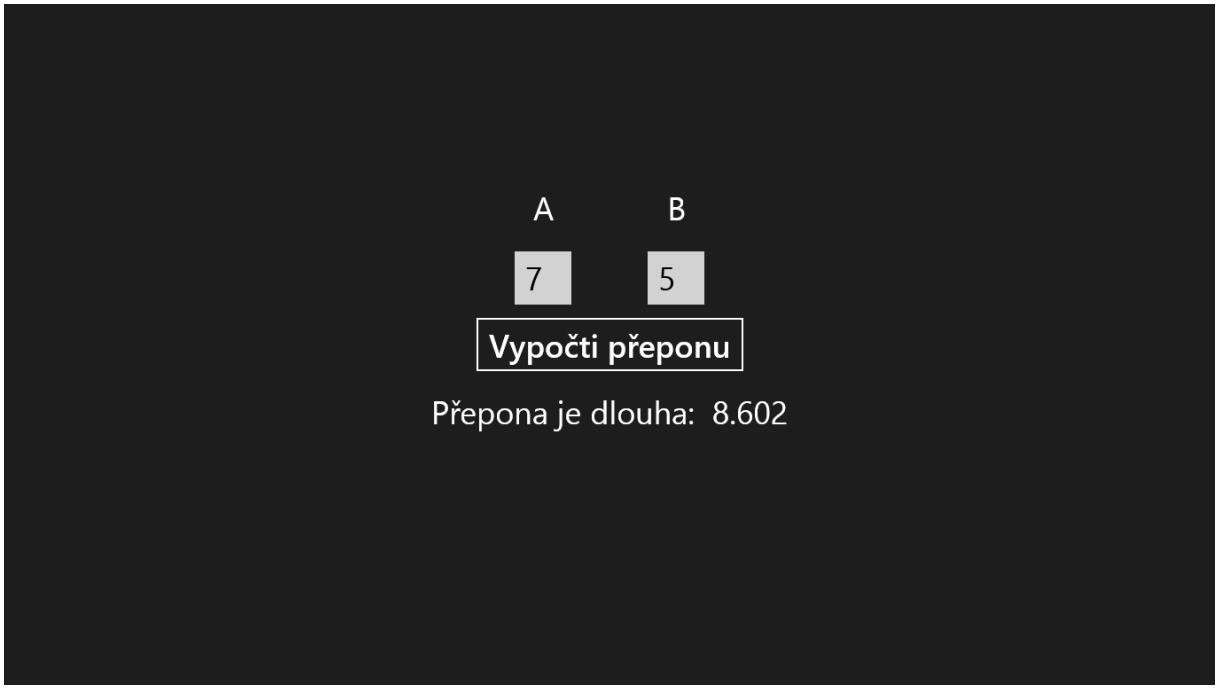
```
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">

        <Button Content="Vypočti přeponu" HorizontalAlignment="Center"
  VerticalAlignment="Center" FontSize="36" Click="Button_Click_1" />

        <TextBlock  Name="Ctb" HorizontalAlignment="Center"
  VerticalAlignment="Center" FontSize="36" Margin="0,160,0,0"
  TextWrapping="Wrap" Text=" " />

<TextBlock  HorizontalAlignment="Center" VerticalAlignment="Center"
 FontSize="36" Margin="0,-300,0,0" TextWrapping="Wrap" Text="A           B"/>

 <TextBox Name="Btb" HorizontalAlignment="Center"
  VerticalAlignment="Center" Margin="150,-150,0,0" FontSize="36"
  TextWrapping="Wrap" Text="0" />

        <TextBox Name="Atb" HorizontalAlignment="Center" VerticalAlignment="Center"
         Margin="-150,-150,0,0" FontSize="36" TextWrapping="Wrap" Text="0" />

</Grid>
```

5. Dvakrát klikněte na Tlačítko „Vypočti“ a otevře se záložka MainPage.xaml.cs.

6. Upravte program:

```
private void Button_Click_1(object sender, RoutedEventArgs e)
        {

            // založíme proměnnou A datového typu double
            // zkonvertujeme vlastnost Text z TextBoxu s názvem Atb do double
            // a uložíme do proměnné A

            double A = Convert.ToDouble(Atb.Text);

            // totéž co řádek předtím, a uložíme do proměnné B

            double B = Convert.ToDouble(Btb.Text);

            // vytvoříme proměnnou C
            // vypočítáme druhou odmocninu z A^2 + B^2 a uložíme do proměnné C

            double C = Math.Sqrt(A*A + B*B);

            // do TexBlocku s názvem Ctb a jeho vlastnosti Text uložíme zaokrouhlenou
            // proměnnou C

            Ctb.Text = "Přepona je dlouha:  " + Math.Round(C,3).ToString();
        }
```

7. Výsledek by měl po spuštění vypadat třeba takto:



Datový typ char

Jedná se o datový typ, do kterého lze uložit jeden znak:

```
char znak = 'A';
```

Po vykonání tohoto řádku bude v proměnné „znak“ uložen znak A.

V počítači mají všechny znaky svoji hodnotu i jako celé číslo. Například znak A má hodnotu 65, proto můžeme napsat:

```
int cislo_znaku = 'A';
```

Spojením znaků a čísel vznikne tabulka, které se říká ASCII (American Standard Code for Information Interchange). Její základní část je 7bitová a obsahuje 128 znaků. Protože se do 128 znaků nevejdou národní sady, byla postupně rozšiřována na osm a více bitů. My si ukážeme jen standardních 128 znaků; vynecháme počáteční znaky 0–31, protože se jedná o tzv. neviditelné znaky.

Číslo	Znak	Číslo	Znak	Číslo	Znak	Číslo	Znak
32	space	56	8	80	P	104	H
33	!	57	9	81	Q	105	I
34	"	58	:	82	R	106	J
35	#	59	;	83	S	107	K
36	$	60	<	84	T	108	L
37	%	61	=	85	U	109	M
38	&	62	>	86	V	110	N
39	'	63	?	87	W	111	O
40	(	64	@	88	X	112	P
41	)	65	A	89	Y	113	Q
42	*	66	B	90	Z	114	R
43	+	67	C	91	[	115	S
44	,	68	D	92	\	116	T
45	-	69	E	93	]	117	U
46	.	70	F	94	^	118	V
47	/	71	G	95	_	119	W
48	0	72	H	96	`	120	X
49	1	73	I	97	A	121	Y
50	2	74	J	98	B	122	Z
51	3	75	K	99	C	123	{
52	4	76	L	100	D	124	|
53	5	77	M	101	E	125	}
54	6	78	N	102	F	126	~
55	7	79	O	103	G	127	DEL

ASCII tabulka znaků

Datový typ string

Dalším datovým typem, s kterým se budete ve svých programech často setkávat, je string. Jedná se o datový typ pro ukládání řetězců znaků. Příkladem může být například proměnná jméno, do které uložíme slovo Petr:

```
string jmeno = "Petr";
```

Řetězec znaků může být libovolně dlouhý (omezuje pouze velikost paměti). S použitím řetězců jsme se již setkali v předchozích programech. Například při jakémkoliv výpisu do TextBlocku jsme při ukládání do vlastnosti Text převáděli všechny číselné datové typy do stringu pomocí metody ToString().

Pokud bude potřeba zjistit počet znaků stringu, můžeme použít vlastnost Length:

```
int pocet_znaku = jmeno.Length;
```

V našem případě bude v proměnné pocet_znaku hodnota 4, protože jméno Petr se skládá ze čtyř znaků.

Každý řetězec se skládá ze samostatných znaků, ke kterým se dá přistupovat pomocí indexu. Index je v podstatě ukazatel na jeden znak řetězce. Indexovat se začíná od nuly. Příklad indexů pro náš string můžete vidět v následující tabulce:

Index	0	1	2	3
Znak	P	e	t	r

Index se používá v hranatých závorkách za názvem proměnné. Na následujícím řádku je ukázka proměnné typu char, ve které je uložen znak „e“ z našeho řetězce:

```
char znak = jmeno[1];
```

Datový typ bool

Do proměnných tohoto datového typu lze uložit pouze dvě hodnoty. První je hodnota **true (1)** a znamená pravda a druhá je hodnota **false (0)** a znamená lež. Tento datový typ se používá především pro podmíněné příkazy, o kterých se dozvíme v další kapitole:

```
bool rozhodnuti = true;

bool rozhodnuti = false;
```

Podmíněné příkazy

Podmíněné příkazy se používají všude tam, kde potřebujeme rozdělit běh programu podle nějaké podmínky. Základní příkazem pro porovnání dvou a více proměnných je příkaz if – else.

Názorně si jeho funkci popíšeme na následujícím příkladu. Program řeší, jestli je proměnná A větší než B. Proměnné A a B mohou být libovolného hodnotového typu:

```
// pokud bude A > B, tak se vykoná kód ve složených závorkách pod if
if (A > B)
{
    // zde bude kód, který se vykoná, pokud bude platit podmínka A > B
}
else
{
    // zde bude kód, který se vykoná, pokud nebude platit podmínka A > B
}
```

Pokud bude za if nebo else následovat pouze jeden příkaz, není nutné psát složené závorky. Z vlastní zkušenosti však doporučuji tyto závorky dodržovat, nic tím nepokazíte.

Samozřejmě mezi proměnou A a B můžeme dávat i jiné operátory než „větší“. Jejich seznam uvádí následující tabulka:

Operátor	Popis
>	větší
<	menší
>=	větší nebo rovno
<=	menší nebo rovno
==	rovná se
!=	nerovná se

Program „Největší číslo ze tří“

Než budeme pokračovat, vyzkoušíme si napsat jednoduchý program, který najde největší číslo ze tří zadaných.

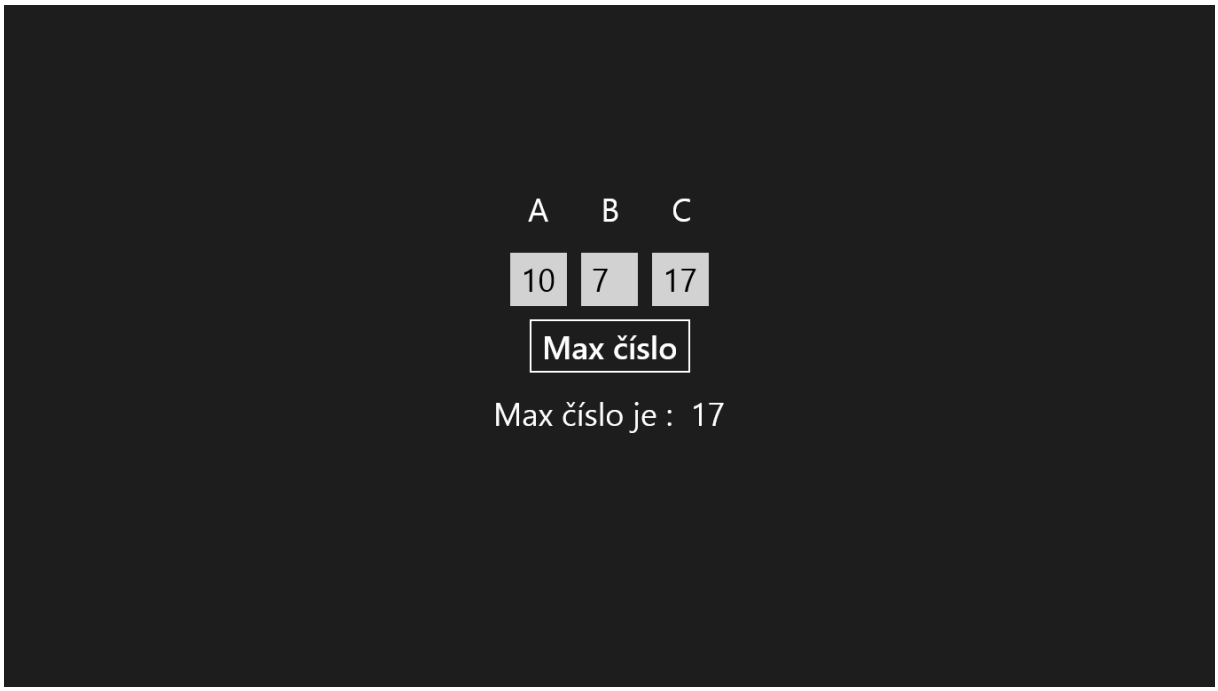
1. Založte nový projekt s názvem MaxCislo.
2. Stejně jako v předešlých programech umístěte komponentu tlačítka (Button) a komponentu TextBlock (pro zobrazení výsledku).
3. Přidejte komponentu TextBox (slouží k zadávání textu).
4. Upravte Xaml podle následujícího kódu:

```
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">

        <Button Content="Max číslo" HorizontalAlignment="Center"
   VerticalAlignment="Center" FontSize="36" Click="Button_Click_1" />

        <TextBlock  Name="Vtb" HorizontalAlignment="Center"
    VerticalAlignment="Center" FontSize="36" Margin="0,160,0,0"
    TextWrapping="Wrap" Text=" " />

        <TextBlock  HorizontalAlignment="Center" VerticalAlignment="Center"
   FontSize="36" Margin="0,-300,0,0" TextWrapping="Wrap" Text="A      B      C"/>

        <TextBox Name="Atb" HorizontalAlignment="Center"
    VerticalAlignment="Center" Margin="160,-150,0,0" FontSize="36"
    TextWrapping="Wrap" Text="0" MaxLength="3"/>

        <TextBox Name="Btb" HorizontalAlignment="Center" VerticalAlignment="Center"
         Margin="0,-150,0,0" FontSize="36" TextWrapping="Wrap" Text="0"
  MaxLength="3"/>

        <TextBox Name="Ctb" HorizontalAlignment="Center" VerticalAlignment="Center"
         Margin="-160,-150,0,0" FontSize="36" TextWrapping="Wrap" Text="0"
     MaxLength="3" />

</Grid>
```

5. Dvakrát klikněte na Tlačítko „Vypočti“ a otevře se záložka MainPage.xaml.cs.
6. Upravte program:

```
private void Button_Click_1(object sender, RoutedEventArgs e)
        {

            // založíme proměnnou A datového typu int
            // zkonvertujeme vlastnost Text z TextBoxu a uložíme do A
            int A = Convert.ToInt32(Atb.Text);

            // totéž co řádek předtím, a uložíme do proměnné B

            int B = Convert.ToInt32(Btb.Text);

            // totéž co řádek předtím, a uložíme do proměnné C

            int C = Convert.ToInt32(Ctb.Text);

            // pokud B bude větší než A, tak se do A uloží hodnota B
            if (B > A)
            {
                A = B;
            }

            // pokud C bude větší než A, tak se do A uloží hodnota C
            if (C > A)
            {
                A = C;
            }

            // zobrazíme proměnnou A, ve které je největší hodnota
            Vtb.Text = "Max číslo je :  " + A.ToString();
        }
```

7. A toto je výsledek:



Složené podmínky

Příkaz if může mít i složitější konstrukci než porovnání mezi dvěma proměnnými. Může nastat případ, kdy budeme potřebovat vytvořit tzv. složenou podmínku. Ukážeme si to na následujícím příkladu.

Obvykle jednou za čtyři roky nastává přestupný rok, kdy počet dnů vzroste na 366. Měsíc únor má v přestupném roce 29 dní. Není tomu však vždy, a proto jsou stanoveny následující podmínky: Rok je přestupný tehdy, pokud je jeho letopočet beze zbytku dělitelný číslem 4 a zároveň není dělitelný 100, nebo je beze zbytku dělitelný číslem 400.

Zkusme vytvořit tuto podmínku pomocí složeného příkazu if – else.

Pokud budeme potřebovat, aby příkaz if obsahoval více než jednu podmínku, musí mezi jednotlivými podmínkami existovat vztah. Tento vztah musí být jednoznačný a jeho výsledkem musí být stejně jako u podmínek pouze True nebo False. Máme dva základní vztahy AND, OR.

AND – a zároveň (&&):

```
if(podminka1 && podminka2)
```

Pokud bude platit podmínka1 a zároveň podmínka2, tak celý if platí. Pokud jedna z podmínek platit nebude, tak celý if neplatí.

OR – nebo (||):

```
if(podminka1 || podminka2)
```

Pokud platí podmínka1 nebo podmínka2, pak celý if platí. Jen v případě, že neplatí žádná z podmínek, tak celý if neplatí.

Také je možné jednotlivé podmínky negovat.

Negace – obrácená hodnota (!):

```
if(!podminka1)
```

Pokud podmínka1 bude platit, tak celý if neplatí, a naopak pokud podmínka1 neplatí, tak celý if platí. Vykřičník otočí (neguje) platnost.

Teď se vrátíme k našemu přestupnému roku. Nejprve je zde otázka, jak zjistíme, že je rok letopočet beze zbytku dělitelný? Stačí použít operátor modulo, a pokud se zbytek po dělení bude rovnat 0, pak je letopočet dělitelný beze zbytku.

Teď už můžeme tvořit naši podmínku. Rok je přestupný tehdy, pokud je jeho letopočet beze zbytku dělitelný číslem 4 a zároveň není dělitelný 100:

```
if((rok%4 == 0)&&(rok%100 != 0))
```

Stačí už jen přidat druhou část „nebo je beze zbytku dělitelný číslem 400“:

```
if((rok%4 == 0)&&(rok%100 != 0)||(rok%400 == 0))
```

Hotovo! Pokud tato podmínka bude platit, je rok přestupný. Ve třídě DateTime už samozřejmě existuje metoda IsLeapYear, která zjišťuje totéž, ale to bychom si tak pěkně neprocvičili složené podmínky…:

```
if(DateTime.IsLeapYear(rok))
```

A teď napíšeme program pro zjišťování přestupného roku.

Program „Výpočet přestupného roku“

1. Založte nový projekt s názvem PrestupnýRok.
2. Stejně jako v předešlých programech umístěte komponentu tlačítka (Button) a komponentu TextBlock (pro zobrazení výsledku).
3. Přidejte komponentu TextBox (slouží k zadávání textu).
4. Upravte Xaml podle následujícího kódu:

```
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">

        <Button Content="Zjisti rok" HorizontalAlignment="Center"
         VerticalAlignment="Center" FontSize="36" Click="Button_Click_1" />

        <TextBlock  Name="Vtb" HorizontalAlignment="Center"
         VerticalAlignment="Center" FontSize="36" Margin="0,160,0,0"
         TextWrapping="Wrap" Text=" " />

         <TextBox Name="BtRok" HorizontalAlignment="Center" VerticalAlignment="Center"
         Margin="0,-150,0,0" FontSize="36" TextWrapping="Wrap" Text="0"
         MaxLength="4"/>


</Grid>
```

5. Dvakrát klikněte na Tlačítko „Zjisti rok“ a otevře se záložka MainPage.xaml.cs, nebo stačí označit myší `Click="Button_Click_1"` a pravým tlačítkem vybrat možnost „Navigate to Event Handler“.

6. Upravte program:

```
private void Button_Click_1(object sender, RoutedEventArgs e)
        {

            int rok = Convert.ToInt32(BtRok.Text);

            //if(DateTime.IsLeapYear(rok))

            if ((rok % 4 == 0) && (rok % 100 != 0) || (rok % 400 == 0))
            {
                Vtb.Text = "Rok " + rok.ToString() + " je přestupný";
            }
            else
            {
                Vtb.Text = "Rok " + rok.ToString() + " není přestupný";
            }


        }
```

7. A takto by měl program vypadat po spuštění:

Už jste někdy programovali hru? V následující kapitole se do toho pustíme.

Hra „Hádání čísla“

Smyslem hry bude uhádnout na deset pokusů náhodně vygenerované číslo od 1 do 100.

Generování náhodného čísla

Nejprve si musíme říct, jak náhodně vygenerovat číslo. Generování něčeho náhodného je pro počítač celkem náročný úkol. Existuje mnoho algoritmů, které se snaží problém náhody řešit, ale jejich matematická obtížnost je vcelku značná. Naštěstí máme k dispozici třídu Random, která se o to postará:

```
// vytvoříme objekt nahoda z třídy Random
Random nahoda = new Random();

// do int nahodne_cislo uložíme náhodné číslo vygenerované metodou Next
// v rozsahu 1-100
int nahodne_cislo = nahoda.Next(1, 101);
```

Je to jednoduché a můžeme se pustit do programování naší první hry.

1. Založte nový projekt s názvem HadaniCisla.
2. Stejně jako v předešlých programech umístěte komponenty Button, TextBlock a TextBox.
3. Upravte Xaml podle následujícího kódu:

```
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">

        <Button Content="Potvrdit" HorizontalAlignment="Center"
         VerticalAlignment="Center" FontSize="36" Click="Button_Click_1" />

        <TextBlock  HorizontalAlignment="Center"
         VerticalAlignment="Center" FontSize="36" Margin="0,-200,0,0"
         TextWrapping="Wrap" Text="Zadej číslo na které myslím" />

        <TextBlock  Name="Tbpokus" HorizontalAlignment="Center"
         VerticalAlignment="Center" FontSize="36" Margin="0,-320,0,0"
         TextWrapping="Wrap" Text="Zbývá 10 pokusů" />

        <TextBlock  Name="Vtb" HorizontalAlignment="Center"
         VerticalAlignment="Center" FontSize="36" Margin="0,160,0,0"
         TextWrapping="Wrap" Text=" " />

<TextBox Name="Tbcislo" HorizontalAlignment="Center"
 VerticalAlignment="Center" Margin="-300,0,0,0" FontSize="36" TextWrapping="Wrap"
Text="0" MaxLength="3"/>

</Grid>
```

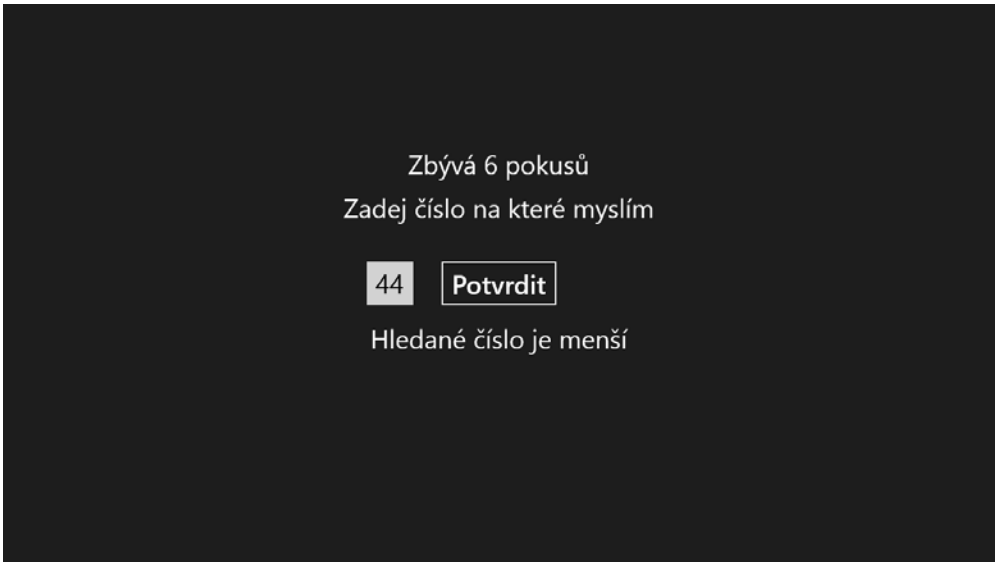
4. Dvakrát klikněte na Tlačítko „Potvrdit“ a otevře se záložka MainPage.xaml.cs, nebo stačí označit myší `Click="Button_Click_1"` a pravým tlačítkem vybrat možnost „Navigate to Event Handler“.

5. Opište následující program:

```
public MainPage()
{
    this.InitializeComponent();

    // při spuštění programu uložíme do proměnné nahodne_cislo náhodné
    // číslo vygenerované metodou Next v rozsahu 1-100
    nahodne_cislo = nahoda.Next(1, 101);
}


// vytvoříme objekt nahoda z třídy Random
Random nahoda = new Random();

// založíme proměnné
int nahodne_cislo;
int pocet_pokusu = 0;
bool nova_hra = false;


private void Button_Click_1(object sender, RoutedEventArgs e)
{

    // připočteme pokus
    pocet_pokusu++;

    // kontrola počtu pokusů, více než 10 prohra
    if (pocet_pokusu <= 10)
    {

        // zobrazíme počet zbývajících pokusů
        Tbpokus.Text = string.Format("Zbývá {0} pokusů", 10-pocet_pokusu);

        // do proměnné číslo uložíme zadanou hodnotu
        int cislo = Convert.ToInt16(Tbcislo.Text);

        // pokud bude zadané číslo větší
        if (cislo > nahodne_cislo)
        {
            // vypíše se
            Vtb.Text = "Hledané číslo je menší";
        }

        // pokud bude zadané číslo menší
        if (cislo < nahodne_cislo)
        {
            // vypíše se
            Vtb.Text = "Hledané číslo je větší";
        }
```

```
            // pokud se bude shodovat
            if (cislo == nahodne_cislo)
            {
                // vypíše se text
                Vtb.Text = "Vyhrál jsi";

                // nastavíme příznak nové hry
                nova_hra = true;
            }
        }
        else
        {
            // zobrazíme prohru
            Vtb.Text = "Bohužel jsi prohrál, hledané číslo bylo: "+
                        nahodne_cislo.ToString();

            // nastavíme příznak nové hry
            nova_hra = true;

        }

        // pokud má proměnná nova_hra hodnotu true
        if (nova_hra)
        {
            // znovu vygenerujeme nové číslo 1-100
            nahodne_cislo = nahoda.Next(1, 101);

            //vynulujeme počet pokusů
            pocet_pokusu = 0;

            // zobrazíme počet pokusů
            Tbpokus.Text = "Zbývá 10 pokusů";

            // vrátíme zpět příznak nové hry
            nova_hra = false;
        }

    }
```

6. Výsledek bude vypadat třeba takto:



Příkaz Switch

V případě, že budete potřebovat rozdělit běh programu na více než dvě části, je vhodné použít příkaz Switch. Switch znamená anglicky přepínač a stejně tak se chová i v programu. Základní konstrukce vypadá následovně:

```
// do proměnné A uložíme nějaké číslo, podle kterého budeme přepínat
int A = ?;

// výstupní řetězec
string retez;

// vstupní proměnná A, podle které se bude přepínat
switch (A)
{
    // za příkazem case následuje číslo, se kterým se porovnává
    // hodnota v proměnné A
    // pokud se hodnoty rovnají, vykonají se příkazy za dvojtečkou
    // pokud program dojde k příkazu break, ukončí se celý switch
    // pokud není za dvojtečkou žádný příkaz, tak se program propadne
    // za následující case a pokračuje, dokud nenarazí na break

    case 5:
    case 10: retez = "A se rovná 10 nebo 5"; break;
    case 20: retez = "A se rovná 20"; break;
    case 30: retez = "A se rovná 30"; break;
    case 40: retez = "A se rovná 40"; break;

    // pokud se hodnota proměnné nerovná žádné hodnotě za case,
    // vykoná se následující řádek
   default: retez = „A se nerovná se žádnému číslu“; break;

}
```

Je nutné si uvědomit, že pokud budeme potřebovat, aby se jednotlivé větve programu propadaly, nesmí za dvojtečkou následovat žádný příkaz.

V následujícím programu si prakticky ukážeme použití Switche. Program bude po zadání čísla měsíce vypisovat, kolik má daný měsíc dnů. Zadáme-li například 3 (březen), program vypíše, že má měsíc 31 dní.

Tabulka měsíců a počet jejich dnů:

Číslo měsíce	1	2	3	4	5	6	7	8	9	10	11	12
Počet dnů	31	28–29	31	30	31	30	31	31	30	31	30	31

Program „Kolik má měsíc dnů“

1. Založte nový projekt s názvem PocetDnu.
2. Stejně jako v předešlých programech umístěte komponentu tlačítka (Button) a komponentu TextBlock (pro zobrazení výsledku).
3. Přidejte komponentu TextBox (slouží k zadávání textu).
4. Upravte Xaml podle následujícího kódu:

```
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">

        <Button Content="Zjisti počet dnů" HorizontalAlignment="Center"
         VerticalAlignment="Center" FontSize="36" Click="Button_Click_1" />

        <TextBlock  HorizontalAlignment="Center"
         VerticalAlignment="Center" FontSize="36" Margin="0,-300,0,0"
         TextWrapping="Wrap" Text="Zadej číslo měsíce" />


        <TextBlock  Name="Vtb" HorizontalAlignment="Center"
         VerticalAlignment="Center" FontSize="36" Margin="0,160,0,0"
         TextWrapping="Wrap" Text=" " />

<TextBox Name="Tbmesic" HorizontalAlignment="Center"
 VerticalAlignment="Center" Margin="0,-150,0,0" FontSize="36"
 TextWrapping="Wrap" Text="0" MaxLength="2"/>


</Grid>
```

5. Dvakrát klikněte na Tlačítko „Zjisti počet dnů“ a otevře se záložka MainPage.xaml.cs, nebo stačí označit myší `Click="Button_Click_1"` a pravým tlačítkem vybrat možnost „Navigate to Event Handler“.

6. Než napíšete program, je třeba si uvědomit, jak co nejlépe vytvořit strukturu Switche. Můžete napsat dvanáct příkazů Case s hodnotami 1–12, ale existuje daleko jednodušší způsob. Pokud se pozorně podíváte na tabulku měsíců, zjistíte, že nejčastější (7×) jsou měsíce, které mají 31 dní. Následují měsíce s 30 dny (4×) a nakonec únor, který má 28 nebo 29 dní, podle přestupného roku. Vzhledem k tomu, že nezadáváme letopočet, budeme v našem programu počítat s 28 dny. Využijeme propadávání, což celý program zjednoduší.

7. Další možností je využít default pro měsíce, jež mají 31 dní, ale potom se vyplatí udělat před příkazem Switch kontrolu čísla měsíce, jinak by program po zadání jiného čísla než 1–12 psal, že má tento měsíc 31 dní.

8. Napište následující program:

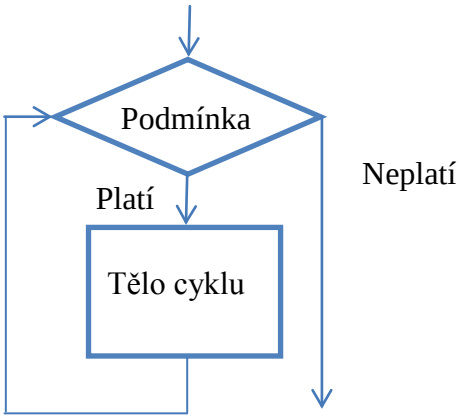
```
private void Button_Click_1(object sender, RoutedEventArgs e)
        {
            // do proměnné M uložíme číslo měsíce
            int M = Convert.ToInt16(Tbmesic.Text);

            // založíme proměnnou pocetdnu datového typu string
            string pocetdnu;

            // kontrolujeme rozsah proměnné M, jestli je mezi 1 a 12
            if (M >= 1 && M <= 12)
            {

                // vstupní proměnná M, podle které se bude přepínat
                switch (M)
                {
                    // pro 4, 6, 9, 11 se do proměnné pocetdnu uloží 30
                    case 4:
                    case 6:
                    case 9:
                    case 11: pocetdnu = "Měsíc má 30 dní"; break;
                    // pro únor 28
                    case 2: pocetdnu = "Měsíc má 28 dní"; break;
                    // ostatní měsíce mají 31 dní
                    default: pocetdnu = "Měsíc má 31 dní"; break;
                }
            }

            // v případě špatného rozsahu se vypíše
            else
            {
                pocetdnu = "Špatné zadaní, zadej číslo měsíce od 1-12";
            }

            // do TexBlocku s názvem Vtb a jeho vlastnosti Text uložíme pocetdnu
            Vtb.Text = pocetdnu;
        }
```

9. Toto je možný výsledek:

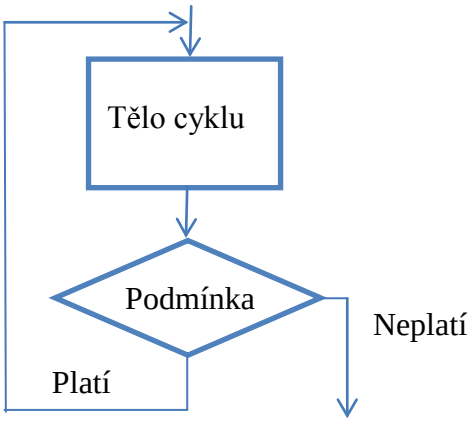
Cykly

Pokud to s programováním myslíte opravdu vážně, tak se bez cyklů neobejdete. Cyklus umožní část kódu několikrát opakovat. Část kódu, která se opakuje, se nazývá tělo cyklu. Cyklů je celá řada, ale v základě je můžeme rozdělit následovně:

1. **Cykly s podmínkou na začátku** – jedná se o velmi využívaný typ cyklu, kde nejprve dochází ke kontrole podmínky a v případě, že podmínka platí, dojde k průchodu tělem cyklu. U těchto cyklů může nastat případ, že tělo cyklu neproběhne ani jednou.



2. **Cykly s podmínkou na konci** – při průchodu programu se nejprve provede tělo cyklu a posléze se testuje, jestli se tělo bude opakovat, nebo program poběží dál.



3. **Cykly s předem daným počtem opakování** jsou takové, kdy předem víme, kolikrát se bude cyklus opakovat.

4. **Cykly s neznámým počtem opakování** jsou takové, u nichž nelze počet opakování předem odhadnout. Příkladem jsou zadávací cykly, kdy uživatel nezná počet zadávaných hodnot, ví jen to, že zadávání ukončí zadáním speciálního znaku, hodnoty nebo slova.

Cyklus For

Jedná se o jeden z nejčastěji používaných cyklů. Je to cyklus s podmínkou na začátku a s předem daným počtem opakování. Jeho definice se skládá ze tří částí oddělených středníkem.

1. **Inicializační část** – v této části je možné založit proměnnou a nastavit její počáteční hodnotu.
2. **Podmínka** – v této části cyklu se vyhodnocuje podmínka. Pokud podmínka platí, cyklus se vykoná.
3. **Inkrementační/dekrementační část** slouží ke změně hodnoty proměnné vytvořené v inicializační části.

```
// v inicializační části je založena proměnná i a do ní je uložena hodnota 0
// je testována podmínka, jestliže je i menší než 10, tak se cyklus vykoná
// v inkrementační části je proměnná i zvětšena při každém průchodu o 1
// cyklus projde tělo 10x, proměnná i se bude postupně zvětšovat od 0 do 9
// pokud bude i rovno 10, přestane platit podmínka a cyklus se ukončí

            for (int i = 0; i < 10; i++)
            {
                //zde se nachází tělo cyklu
            }
```

Program „Součet sudých čísel“

Abychom si procvičili příkaz for, vytvoříme program, který bude počítat součet sudých čísel v zadaném intervalu. Nejprve je třeba si říct, jak zjistíme, je-li číslo sudé. Stačí vytvořit podmínku, kdy budeme testovat, zda bude celočíselný zbytek po dělení 2 roven 0. Zvolený interval jednoduše zadáme do cyklu for.

1. Založte nový projekt s názvem SoucetSudCisel.
2. Stejně jako v předešlých programech umístěte komponentu tlačítka (Button) a komponentu TextBlock (pro zobrazení výsledku).
3. Přidejte komponentu TextBox (slouží k zadávání textu).
4. Upravte Xaml podle následujícího kódu:

```
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">

        <Button Content="Vypočti" HorizontalAlignment="Center"
         VerticalAlignment="Center" FontSize="36" Click="Button_Click_1" />

        <TextBlock  HorizontalAlignment="Center"
         VerticalAlignment="Center" FontSize="36" Margin="0,-300,0,0"
         TextWrapping="Wrap" Text="Zadej interval čísel od od" />
```

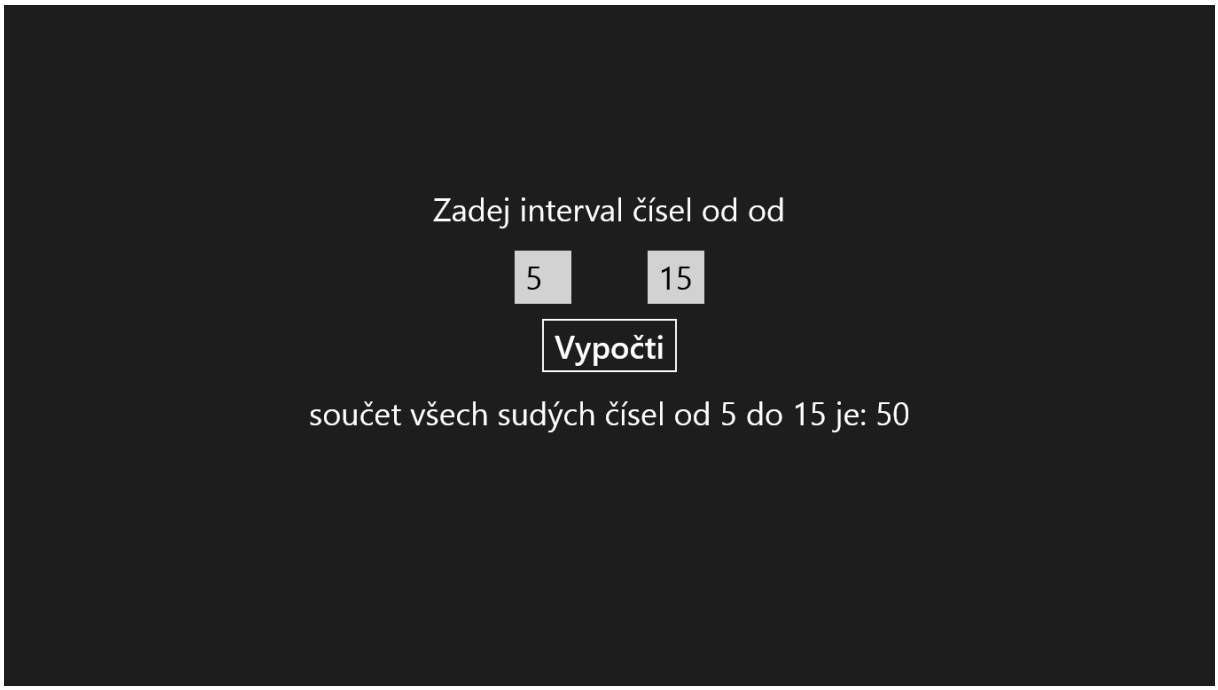
```
        <TextBlock  Name="Vtb" HorizontalAlignment="Center"
         VerticalAlignment="Center" FontSize="36" Margin="0,160,0,0"
         TextWrapping="Wrap" Text=" " />

        <TextBox Name="Tbmin" HorizontalAlignment="Center" VerticalAlignment="Center"
         Margin="-150,-155,0,0" FontSize="36" TextWrapping="Wrap" Text="0"
         MaxLength="4"/>

        <TextBox Name="Tbmax" HorizontalAlignment="Center" VerticalAlignment="Center"
         Margin="150,-155,0,0" FontSize="36" TextWrapping="Wrap" Text="0"
         MaxLength="4"/>

</Grid>
```

5. Dvakrát klikněte na Tlačítko „Vypočti“ a otevře se záložka MainPage.xaml.cs, nebo stačí označit myší `Click="Button_Click_1"` a pravým tlačítkem vybrat možnost „Navigate to Event Handler“.

6. Opište následující program:

```
private void Button_Click_1(object sender, RoutedEventArgs e)
        {

            // do proměnné min uložíme dolní mez intervalu
            int min = Convert.ToInt16(Tbmin.Text);

            // do proměnné max uložíme horní mez intervalu
            int max = Convert.ToInt16(Tbmax.Text);

            // založíme proměnnou součet a uložíme do ní hodnotu 0, je to nutné kvůli
            // přičítání
            int soucet = 0;

            // vytvoříme cyklus, kde se proměnná i postupně mění od min do max
            for (int i = min; i <= max; i++)
            {
                // tento if zjišťuje, jestli se jedná o sudé číslo
                if (i % 2 == 0)
                {
                    // zde přičítáme sudé číslo do proměnné součet
                    soucet = soucet + i;
                }
            }

            // do TexBlocku s názvem Vtb a jeho vlastnosti Text uložíme formátovaný
            // string
            // metodu Format je dobré používat tam, kde je potřeba zobrazovat více
            // proměnných společně s textem
            // první parametr je řetězec a na místech složených závorek s čísly se
            // budou později zobrazovat proměnné
            // {0} - min,{1} - max, {2} - soucet

    Vtb.Text = string.Format("součet všech sudých čísel od {0} do {1} je: {2}",
    min, max, soucet);
        }
```

7. Zde je výsledek naší snahy:



Cyklus While

While je cyklus s podmínkou na začátku, ale na rozdíl od For není předem dán počet opakování. Tento cyklus nemá tzv. čítač a počet opakování se určuje na základě platnosti podmínky cyklu:

```
// cyklus se vykonává, dokud je splněna podmínka
while (podminka)
{
    // zde je tělo cyklu
}
```

Pokud budeme chtít vykonat desetkrát průchod tělem cyklu, bude to vypadat takto:

```
int A = 0;

// cyklus se vykonává, dokud je splněna podmínka
while (A < 10)
{
    // tady zvětšíme A o 1
    A++;
}
```

Zkusíme upravit předchozí příklad, jak by vypadal s použitím cyklu While. Upravíme jen samotný kód programu, Xaml bude stejný:

```
private void Button_Click_1(object sender, RoutedEventArgs e)
        {

            // do proměnné min uložíme dolní mez intervalu
            int min = Convert.ToInt16(Tbmin.Text);

            // do proměnné max uložíme horní mez intervalu
            int max = Convert.ToInt16(Tbmax.Text);

            // založíme proměnnou součet a uložíme do ní hodnotu 0, je to nutné kvůli
            // přičítání
            int soucet = 0;

            // vytvoříme cyklus s podmínkou
            while (min <= max)
            {
                // tento if zjišťuje, jestli se jedná o sudé číslo
                if (min % 2 == 0)
                {
                    // zde přičítáme sudé číslo do proměnné součet
                    soucet = soucet + min;
                }

                // při každém průchodu zvětšíme min o 1
                min++;

            }


     // do TexBlocku s názvem Vtb a jeho vlastnosti Text uložíme formátovaný
     // string

     Vtb.Text = string.Format("součet všech sudých čísel od {0} do {1} je: {2}",
     min, max, soucet);
        }
```

V případě, že jste vše napsali správně, bude program pracovat stejně i s použitím nového cyklu While.

Cyklus Do-While

Do-While je druh cyklu s podmínkou na konci. Jak jsme si řekli v předcházejícím textu, tento druh cyklu provede minimálně jednou průchod tělem cyklu a pak teprve dochází k vyhodnocení podmínky. Tento cyklus se nepoužívá tak často jako předchozí cykly. Zápis cyklu vypadá následovně:

```
            // cyklus se vykonává, dokud je splněna podmínka
            do
            {

            } while (podminka);
```

Pokud budeme chtít vykonat desetkrát průchod tělem cyklu, bude to vypadat takto:

```
int A = 0;

        // cyklus se vykonává, dokud je splněna podmínka
        do
        {
            // tady zvětšíme A o 1
            A++;
        } while (A < 10);
```

Všimněte si několika zajímavých věcí:

1. Za cyklem se píše středník (středník slouží k ukončení příkazu).
2. Proměnná A je zvětšena o 1 před podmínkou.

Máme za sebou základní cykly, zbývá nám už jen jeden, ale ten si necháme na později.

Jednorozměrné pole

Jednorozměrné pole je datová struktura pro uchovávání většího množství dat stejného datového typu. Je to jednoduché: Představte si, že v paměti počítače máte za sebou uloženy proměnné stejného datového typu. Jednotlivé proměnné nemají přiřazeno žádné jméno, jen celá skupina se nějak jmenuje. Pokud je celá skupina proměnných pojmenována jedním názvem, nastává problém, jak pracovat s konkrétní proměnnou. Není to nic těžkého – stejně jako u stringů nám pomohou indexy.

Založení pole

Jednorozměrné pole si založíte třeba takto:

```
        // vytvoří se jednorozměrné pole typu integer s názvem pole1
        // velikost pole je 6 prvků, do pole1 se uloží hodnoty 8, 7, 6, 2, 1, 9
        int[] pole1 = { 8, 7, 6, 2, 1, 9 };
```

V paměti si ho můžete představit takto:

Index pole [ ]	0	1	2	3	4	5
Hodnota pole	8	7	6	2	1	9

V případě, že budete potřebovat vytvořit prázdné pole o určitém rozměru, použijte následující zápis:

```
        // pole má velikost 10 prvků datového typu integer a je vyplněno nulami
        int[] pole1 = new int[10];
```

Velikost pole

Pole jsou indexována od nuly. V případě, že budeme potřebovat zjistit celkovou velikost pole, stačí použít vlastnost Length:

```
int velikost = pole1.Length;
```

V našem případě bude v proměnné velikost hodnota 6.

Čtení z pole

Můžeme přečíst libovolnou položku pole:

```
int polozka = pole1[1];
```

V proměnné polozka bude hodnota 7.

Zápis do pole

Můžeme přepsat libovolnou položku pole:

```
pole1[4] = 16;
```

Pole bude potom vypadat takto:

Index pole [ ]	0	1	2	3	4	5
Hodnota pole	8	7	6	2	16	9

Setřídění pole

Pole lze jednoduše setřídit vzestupně. Použijeme na to třídu Array a metodu Sort():

```
Array.Sort(pole1);
```

Pole po setřídění vypadá takto:

Index pole [ ]	0	1	2	3	4	5
Hodnota pole	2	6	7	8	9	16

Otočení pole

Ještě můžeme pole reverzovat (otočit). Použijeme na to stejnou třídu Array a metodu Reverse():

```
Array.Reverse(pole1);
```

Pole po otočení vypadá takto:

Index pole [ ]	0	1	2	3	4	5
Hodnota pole	16	9	8	7	6	2

Výpis pole

Pro výpis pole použijeme cyklus. Budeme procházet jednotlivá políčka, dokud nedojdeme na konec pole. Nejprve použijeme cyklus For a později si ukážeme i nový cyklus Foreach.

1. Upravte Xaml:

```
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
        <Button Content="Vypiš pole"
   HorizontalAlignment="Center"
   VerticalAlignment="Center"
   FontSize="36" Click="Button_Click" />

        <TextBlock HorizontalAlignment="Center"
   VerticalAlignment="Center"
         Name="Vypis"
   FontSize="36"
   Margin="0,160,0,0"
   TextWrapping="Wrap" />

</Grid>
```

2. Napište program:

```
private void Button_Click(object sender, RoutedEventArgs e)
        {
            // vytvoří se jednorozměrné pole typu integer s názvem pole1
            // velikost pole je 6 prvků, do pole1 se uloží hodnoty 8, 7, 6, 2, 1, 9
            int[] pole1 = { 8, 7, 6, 2, 1, 9 };

            // cyklus prochází postupně celé pole
            for (int i = 0; i < pole1.Length; i++)
            {
                // do vlastnosti text se přidávají jednotlivé hodnoty z pole
                Vypis.Text = Vypis.Text + string.Format("{0},",pole1[i]);
            }


        }
```

Cyklus Foreach

Cyklus Foreach se používá na výpis datových kolekcí. Mezi datové kolekce patří i jednorozměrné pole. Jeho použití je velice snadné a nebudeme potřebovat pracovat s indexy. Foreach prochází postupně celé pole a sám si hlídá počet průchodů podle jeho velikosti. Použití si ukážeme na předchozím příkladu výpisu pole:

```
private void Button_Click(object sender, RoutedEventArgs e)
        {
            // vytvoří jednorozměrné pole typu integer s názvem pole1
            // velikost pole je 6 prvků, do pole1 se uloží hodnoty 8, 7, 6, 2, 1, 9
            int[] pole1 = { 8, 7, 6, 2, 1, 9 };

            // cyklus prochází celé pole a do proměnné polozka ukládá postupně hodnoty
            foreach (int polozka in pole1)
            {
                // do vlastnosti text se přidávájí jednotlivé hodnoty z pole
                // prom1 += prom2 je zkrácený zápis prom1 = prom1 + prom2
                Vypis.Text += string.Format("{0},",polozka);
            }


        }
```

Program „Překladač do Morseovy abecedy “

V Morseově abecedě je každý znak interpretován jako skupina symbolů složená z čárek a teček (krátký a dlouhý signál). Tuto abecedu navrhnul už na začátku 19. století americký malíř a vynálezce Samuel Morse. Dodnes se tato abeceda pro svoji jednoduchost používá především v nouzových stavech. Náš program bude pro zjednodušení kódovat pouze znaky A–Z:

A	.-	H	....	O	---	V	...-
B	-...	I	..	P	.--.	W	.--
C	-.-.	J	.---	Q	--.-	X	-..-
D	-..	K	-.-	R	.-.	Y	-.--
E	.	L	.-..	S	...	Z	--..
F	..-.	M	--	T	-		
G	--.	N	-.	U	..-		

Tabulka Morseovy abecedy

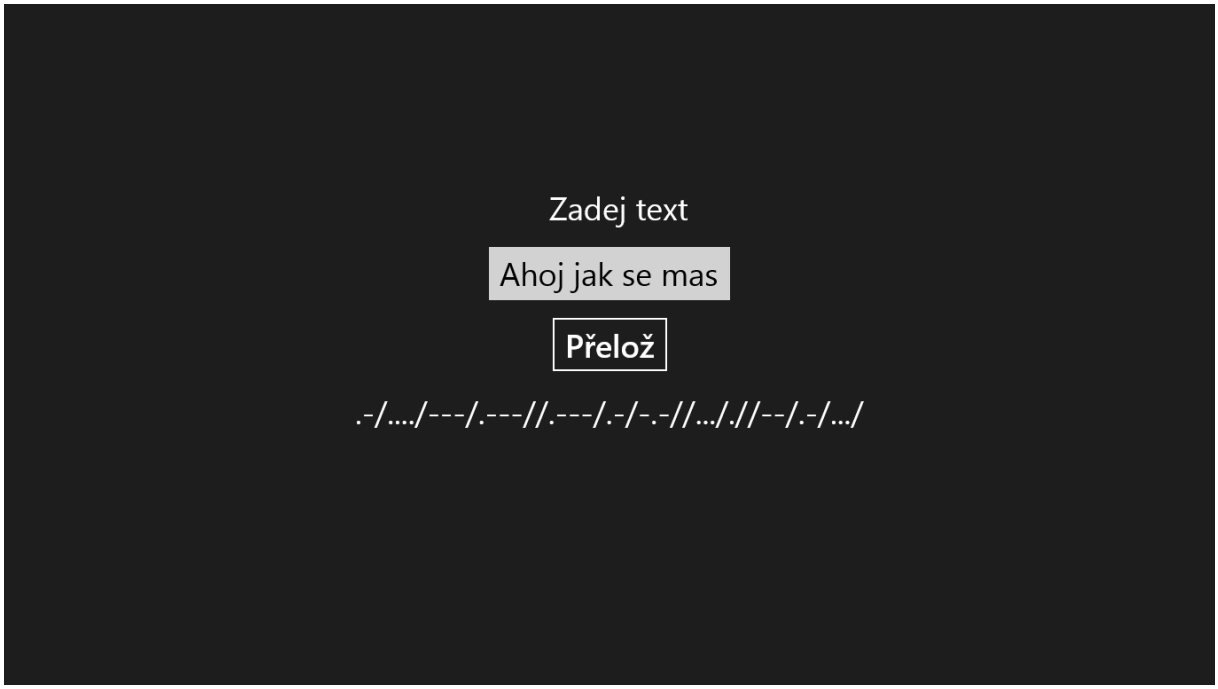
1. Založte projekt s názvem PrekladacMorse.
2. Upravte Xaml:

```
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">

        <Button Content="Přelož" HorizontalAlignment="Center"
         VerticalAlignment="Center" FontSize="36" Click="Button_Click" />

        <TextBlock  Name="Vysledek" HorizontalAlignment="Center"
         VerticalAlignment="Center" FontSize="36" Margin="0,160,0,0"
         TextWrapping="Wrap" Text=" " />

        <TextBlock  HorizontalAlignment="Center" VerticalAlignment="Center"
         FontSize="36" Margin="20,-300,0,0" TextWrapping="Wrap" Text="Zadej text"/>

        <TextBox Name="Vstup" HorizontalAlignment="Center"
         VerticalAlignment="Center" Margin="0,-160,0,0" FontSize="36"
         TextWrapping="Wrap" Text="Ahoj" />

</Grid>
```

3. Napište program:

```
private void Button_Click(object sender, RoutedEventArgs e)
        {
            // pole Morseovy abecedy A ".-" index 0, B "-..." index 1 .....
            string[] pole = { ".-", "-...", "-.-.", "-..", ".", "..-.", "--.",
                              "....", "..", ".---", "-.-", ".-..", "--", "-.",
                              "---", ".--.", "--.-", ".-.", "...", "-", "..-",
                              "...-", ".--", "-..-", "-.--", "--.." };

            // do proměnné zadano se uloží vstupní text a převede se celý na velká
            // písmena
            string zadano = Vstup.Text.ToUpper();

            // přepíšeme předešlý řetězec z vlastnosti Text
            Vysledek.Text = "";

            // založíme pomocnou proměnnou
            string preklad;

            // cyklus prochází vstupní text znak po znaku až do konce textu
            for (int i = 0; i < zadano.Length; i++)
            {
                // pokud je v textu znak mezery
                if (zadano[i] == ' ')
                {
                    // do výstupu se přidá znak lomítka
                    Vysledek.Text +="/";
                }
                // pokud není znak mezery
                else
                {
                    // testuje se, jestli se jedná o znak v rozmezí A-Z
                    if (zadano[i] >= 'A' && zadano[i] <= 'Z')
                    {
                        // do proměnné preklad se uloží znaky Morseovy abecedy z pole
                        // index se vypočítá tak, že od aktuálního znaku odečteme ‚A‘
                        // pokud se podíváte do ASCII tabulky, tak A = 65
```

```
                // například pokud je aktuální znak C (67 ASCII), 67-65 = 2
                // na indexu 2 v „poli“ je v Morseově abecedě "-.-." = C
                preklad = pole[zadano[i] - 'A'];

                // do výstupu přidáme řetězec morseovky pro znak
                Vysledek.Text += string.Format("{0}/", preklad);
            }
            // pokud to není znak A-Z, tak ho nelze přeložit a vypíše se "?"
            else
            {
                Vysledek.Text += string.Format("?/");
            }
        }
    }
}
```



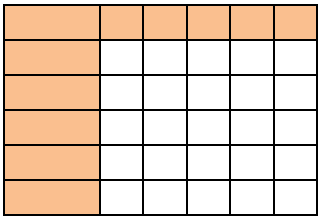
Program po spuštění

Dvourozměrné pole

Stejně jako jednorozměrné pole je toto pole datovou strukturou pro uchování dat stejného datového typu. Přidáním dalšího rozměru je možné si dvourozměrné pole představit jako tabulku dat. K jednotlivým hodnotám v poli se dostaneme pomocí dvou indexů: první index

zpravidla určuje sloupec a druhý řádek tabulky. Pro představu si můžeme ukázat pole o velikosti 5 × 5:

Index	0	1	2	3	4
0	2	4	7	4	6
1	0	1	3	2	7
2	6	0	8	5	9
3	8	9	1	8	0
4	4	2	9	5	2



Příklad pole velikost 5 × 5

Založení 2D pole

Dvourozměrné pole založíme podobně jako jednorozměrné. Pole datového typu integer s již uloženými hodnotami je možné vytvořit následovně:

```
// toto je předvyplněné pole 5x5 datového typu integer
int[,] pole2d = { {2,4,7,4,6},
                  {0,1,3,2,7},
                  {6,0,8,5,9},
                  {8,9,1,8,0},
                  {4,2,9,5,2} };
```

Index	0	1	2	3	4
0	2	4	7	4	6
1	0	1	3	2	7
2	6	0	8	5	9
3	8	9	1	8	0
4	4	2	9	5	2

Pokud budeme potřebovat prázdné pole určitého rozměru, stačí to napsat takto. Pole bude naplněno nulami:

```
// toto je prázdné pole 5x5 datového typu integer
int[,] pole2d2 = new int[5, 5];
```

Index	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0

Zjištění velikosti 2D pole

Pro zjištění celkové velikosti je možné použít jako u jednorozměrného pole vlastnost Length:

```
int velikost = pole2d.Length;
```

V našem případě vrátí hodnotu 25.

Častěji budeme potřebovat informaci o počtu sloupců a řádků. Pro zjištění této informace je dobré použít metodu GetLength(parametr) se vstupním parametrem 0 pro řádky a 1 pro sloupce:

```
// počet řádků
int pocet_radku = pole2d.GetLength(0);

        // počet sloupců
        int pocet_sloupcu = pole2d.GetLength(1);
```

Čtení z 2D pole

Můžeme přečíst libovolnou položku pole:

Index	0	1	2	3	4
0	2	4	7	4	6
1	0	1	3	2	7
2	6	0	8	5	9
3	8	9	1	8	0
4	4	2	9	5	2

```
int polozka = pole2d[3,2];
```

V proměnné polozka bude hodnota 1.

Zápis do 2D pole

Můžeme přepsat libovolnou položku pole:

```
Pole2d[1,3] = 10;
```

Pole bude vypadat takto:

Index	0	1	2	3	4
0	2	4	7	4	6
1	0	1	3	10	7
2	6	0	8	5	9
3	8	9	1	8	0
4	4	2	9	5	2

Výpis 2D pole

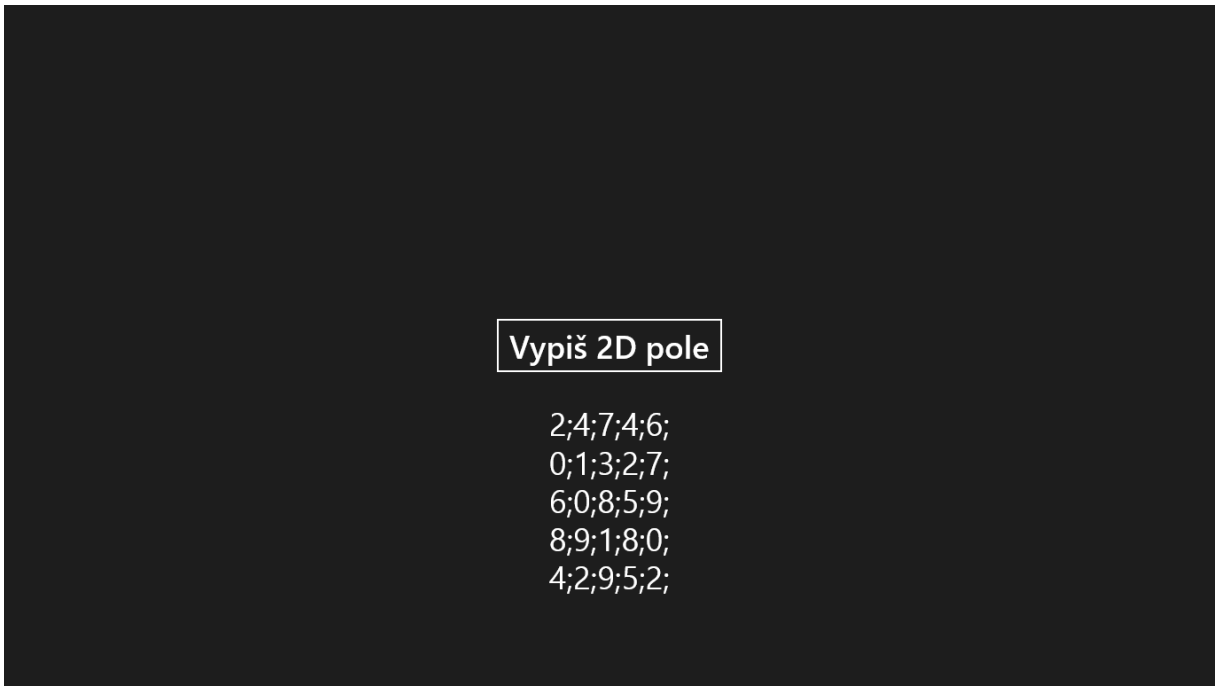
Pokud budeme chtít zobrazit pole jako tabulku, použijeme dva cykly v sobě. První cyklus bude procházet jednotlivé řádky a vnitřní druhý cyklus vždy projde sloupce vybraného řádku. Danou metodu si vyzkoušíme v následujícím příkladu:

1. Upravte Xaml:

```
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
        <Button Content="Vypiš 2D pole"
   HorizontalAlignment="Center"
   VerticalAlignment="Center"
   FontSize="36" Click="Button_Click" />

        <TextBlock HorizontalAlignment="Center"
   VerticalAlignment="Center"
         Name="Vypis"
   FontSize="36"
   Margin="0,400,0,0"
   TextWrapping="Wrap" />

</Grid>
```

2. Napište program:

```
  private void Button_Click(object sender, RoutedEventArgs e)
     {
            // toto je předvyplněné pole 5x5 datového typu integer
            int[,] pole2d = { {2,4,7,4,6},
                              {0,1,3,2,7},
                              {6,0,8,5,9},
                              {8,9,1,8,0},
                              {4,2,9,5,2} };

            // vymazat předchozí string vlastnosti Text
            Vypis.Text = "";

            // tento cyklus prochází řádky
            for (int i = 0; i < pole2d.GetLength(0); i++)
            {
                // v každém řádku projde tento cyklus všechny sloupce
                for (int j = 0; j < pole2d.GetLength(1); j++)
                {
                    // postupně skládá string vlastnosti Text
                    Vypis.Text += string.Format("{0};", pole2d[i, j]);
                }
                // po každém řádku musí odřádkovat (\n skočí na nový řádek)
                Vypis.Text += "\n";
            }

      }
```



Takto bude vypadat výsledek

Hra Lodě

Určitě znáte hru Lodě, v níž je vaším úkolem sestřelit určitý počet skrytých lodí. Vyhráváte ve chvíli, kdy jsou všechny lodě potopeny. Naše hra je záměrně trochu jednodušší. Nenajdete v ní různé typy lodí, ale jen dvourozměrné pole naplněné náhodně číslem 1, což představuje loď, a 0, což představuje vodu. Projděte si následující kód a pak zkuste sami hru vylepšit, třeba o omezený počet pokusů.

1. Upravte Xaml:

```
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
        <Grid.ColumnDefinitions>
            <ColumnDefinition />
            <ColumnDefinition />
        </Grid.ColumnDefinitions>
        <Grid Grid.Column="0">
            <Grid.RowDefinitions>
                <RowDefinition />
                <RowDefinition />
            </Grid.RowDefinitions>

            <Button Width="100" Height="40" Content="Nová hra"
              Click="Zadej_pocet_lodi" HorizontalAlignment="Center"
Margin="0,0,110,0" />
```

```
            <TextBlock Height="20" Width="100" HorizontalAlignment="Center"
             Margin="110,0,0,70" Text="Počet lodí:" FontSize="15" />

            <TextBox Name="Pocet" Height="50" Width="100" HorizontalAlignment="Center"
             Margin="110,0,0,0" />

            <TextBlock Grid.Row="1" Text="Zadej řádek:" Margin="120,0,0,120"
             Height="20" Width="80" FontSize="15" />

            <TextBox Grid.Row="1" Name="Radek" Width="100" Height="50"
             Margin="120,0,0,50" />

            <TextBlock Grid.Row="1" Text="Zadej sloupec:" Margin="0,0,120,120"
             Height="20" Width="100" FontSize="15" />

            <TextBox Grid.Row="1" Name="Sloupec" Width="100" Height="50"
             Margin="0,0,120,50" />

            <Button Grid.Row="1" Width="200" Height="40" Content="Vystřelit"
             Margin="0,60,0,0" HorizontalAlignment="Center" Click="Zadej_souradnice"
             />

            <TextBlock Name="PocetLod" Width="269"  HorizontalAlignment="Center"
             VerticalAlignment="Top" TextWrapping="Wrap" FontSize="26"
             RenderTransformOrigin="0.479,3.831" />

            <TextBlock Name="Info" HorizontalAlignment="Center"  TextWrapping="Wrap"
             Text="Nestřílel" VerticalAlignment="Bottom" FontSize="26"
             RenderTransformOrigin="0.431,2.298" />

        </Grid>
        <TextBlock Grid.Column="1"  Name="Vypis" HorizontalAlignment="Center"
         VerticalAlignment="Center" FontSize="32" />

        <TextBlock  Grid.Column="1"  HorizontalAlignment="Center"
         VerticalAlignment="Top" TextWrapping="Wrap"  FontSize="26" Width="202"
         Height="101" >
             <Run Text="0-Voda"/>
             <LineBreak/>
             <Run Text="1-Zasažená voda"/>
             <LineBreak/>
             <Run Text="2-Potopená loď"/>
        </TextBlock>
</Grid>
```

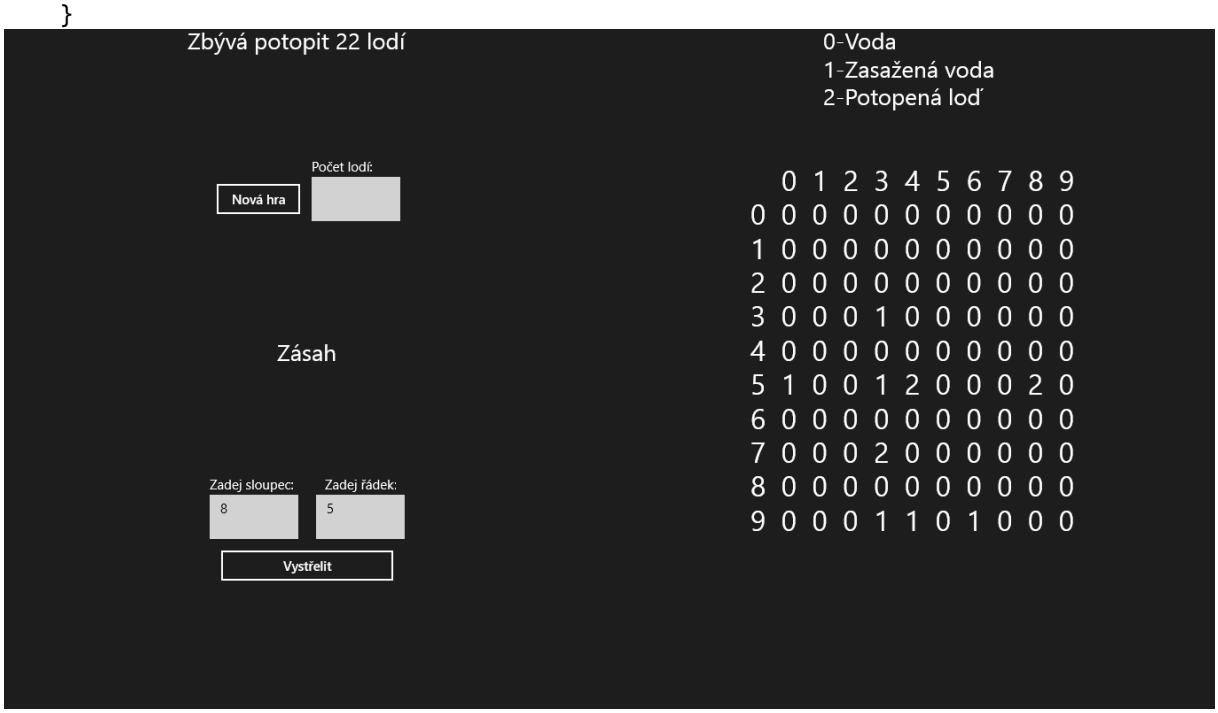
2. Napište program:

```
public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();
        }

        // založíme pole1, do kterého budeme ukládat vygenerované lodě
        int[,] pole1 = new int[10, 10];
        // založíme pole2, které bude sloužit k zobrazování
        char[,] pole2 = new char[10, 10];
```

```
// proměnná lode určuje počet lodí
int lode=25;
// vytvoříme objekt rnd pro generování náhodných čísel
Random rnd= new Random();

// tato metoda se spustí při zapnutí programu
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    // vypíše se, kolik zbývá potopit lodí
    PocetLod.Text = string.Format("Zbývá potopit {0} lodí", lode);

    // postupné skládání stringu hracího pole, toto je první řádek
    Vypis.Text = "    0  1  2  3  4  5  6  7  8  9\n";
    // tyto dva cykly vypíší hrací pole
    for (int i = 0; i < pole1.GetLength(0); i++)
    {
        // výpis čísla řádku
        Vypis.Text += i.ToString()+"  ";

        for (int j = 0; j < pole1.GetLength(1); j++)
        {
            // vynulování a výpis hracího pole
            pole1[i, j] = 0;
            pole2[i, j] = '0';
            Vypis.Text += pole2[i, j]+"  ";
        }
        // odřádkování po každém řádku
        Vypis.Text += "\n";
    }



   // náhodné generování daného počtu lodí
    int a, b;
    for (int i = 0; i < lode; i++)
    {
        // generování indexů a, b
        a = rnd.Next(10);
        b = rnd.Next(10);
        // pokud na vygenerovaném políčku není loď (1), tak
        // se tam uloží
        // pokud tam loď je, prodlouží se cyklus
        if (pole1[a, b] == 0)
            pole1[a, b] = 1;
        else
            i--;
    }

}

// toto je metoda pro kontrolu, co je na políčku, kam střílíme
void Hledej(int a, int b)
{
    // pokud je na políčku v poli1 loď a zároveň
    // v zobrazovacím poli2 není, tak zásah
    if (pole1[a, b] == 1 && pole2[a, b] == '0')
    {
        // oznámení zásahu
        Info.Text = "Zásah";
        // přepsání zobrazovacího pole2 na zásah
        pole2[a, b] = '2';
        // odečtení zbývajících lodí
```

```
            lode--;
        }
        // jinak jsme trefili vodu
        else
        {
            // zobrazení vody v zobrazovacím poli2
            pole2[a, b] = '1';
            // informace o zásahu vody
            Info.Text = "Voda";
        }

        // pokud jsou všechny lodě potopeny, tak vítězství
        if (lode == 0)
        {
            Info.Text = "Vítězství";
        }

        // vypíše se, kolik zbývá potopit lodí
        PocetLod.Text = string.Format("Zbývá potopit {0} lodí", lode);

        // vypíše se zobrazovací pole
        Vypis.Text = "    0  1  2  3  4  5  6  7  8  9\n";
        for (int i = 0; i < pole2.GetLength(0); i++)
        {
            Vypis.Text += i.ToString() + "  ";
            for (int j = 0; j < pole2.GetLength(1); j++)
                Vypis.Text += pole2[i, j] + "  ";
            Vypis.Text += "\n";
        }
    }


    // metoda je volána tlačítkem Výstřel
    private void Zadej_souradnice(object sender, RoutedEventArgs e)
    {
        // kontrola, jestli jsme zadali číslo
        bool numeric = true;

        // cyklus projde zadaný řetězec pro řádek a kontroluje,
        // jestli jsou všechny jeho znaky pouze 0-9
        // pokud ne, tak se proměnná numeric nastaví na false
        for (int i = 0; i < Radek.Text.Length; i++)
        {
            if (!char.IsDigit((char)Radek.Text[i]))
                numeric = false;
        }

        // cyklus projde zadaný řetězec pro sloupec a kontroluje,
        // jestli jsou všechny jeho znaky pouze 0-9
        // pokud ne, tak se proměnná numeric nastaví na false
        for (int i = 0; i < Sloupec.Text.Length; i++)
        {
            if (!char.IsDigit((char)Sloupec.Text[i]))
                numeric = false;
        }

        // pokud jsou všechny znaky čísla a řetězce mají nenulovou velikost
        // a je-li číslo menší než 10 (pole jsou velká 10x10),
        // tak se zavolá metoda Hledej se zadanými indexy
        if (numeric &&
            Radek.Text.Length !=0 &&
            Sloupec.Text.Length !=0 &&
```

```
            Convert.ToInt16(Sloupec.Text) < 10 &&
            Convert.ToInt16(Radek.Text) < 10)
        {
            Hledej(Convert.ToInt16(Radek.Text), Convert.ToInt16(Sloupec.Text));
        }
        // pokud ne, tak je špatné zadání
        else
        {
            Info.Text = "Špatné zadání";
        }
    }

    // metoda volaná tlačítkem nová hra
    private void Zadej_pocet_lodi(object sender, RoutedEventArgs e)
    {
        // kontrola zadání stejné jako u střelby
        bool numeric = true;
        for (int i = 0; i < Pocet.Text.Length; i++)
        {
            if (!char.IsDigit((char)Pocet.Text[i]))
                numeric = false;
        }
        if (numeric && Pocet.Text.Length !=0)
            lode = Convert.ToInt16(Pocet.Text);
        else
        {
            Info.Text = "Špatné zadání";
        }
        // zavolá se metoda OnNavigatedTo jako při spuštění
        OnNavigatedTo(null);
    }
}
```



Takto bude vypadat výsledek

Závěr

Pokud jste dočetli až sem a vyzkoušeli jste všechny programy, nezbývá než vám pogratulovat. Programování je dovednost, která se nezapomíná, a pokud vás bude bavit, tak se postupně budete zlepšovat. V dnešní době máte na výběr v podstatě jen mezi dvěma volbami: buď budete uživateli softwaru, nebo těmi, kdo software programují. Záleží jen na vás, které možnosti dáte přednost. Věžte, že programování je zábava, a využijte možnost naučit se něco nového. Pokud jste studenti, máte v podstatě všechny vývojové nástroje od firmy Microsoft zdarma. Text, který jste právě dočetli, představuje pouze začátek. Doporučuji projít na internetu třeba následující stránky, kde můžete najít další informace a návody:

http://msdn.microsoft.com/cs-cz/windows/apps/br229512.aspx

http://www.devbook.cz/c-sharp-windows-store-aplikace

http://programujte.com/clanek/2013042500-zaciname-vyvijet-aplikace-pro-windows-store-v-html