# Microsoft Small Basic

*Úvod do programování*

*Kapitola 1*

# Úvod

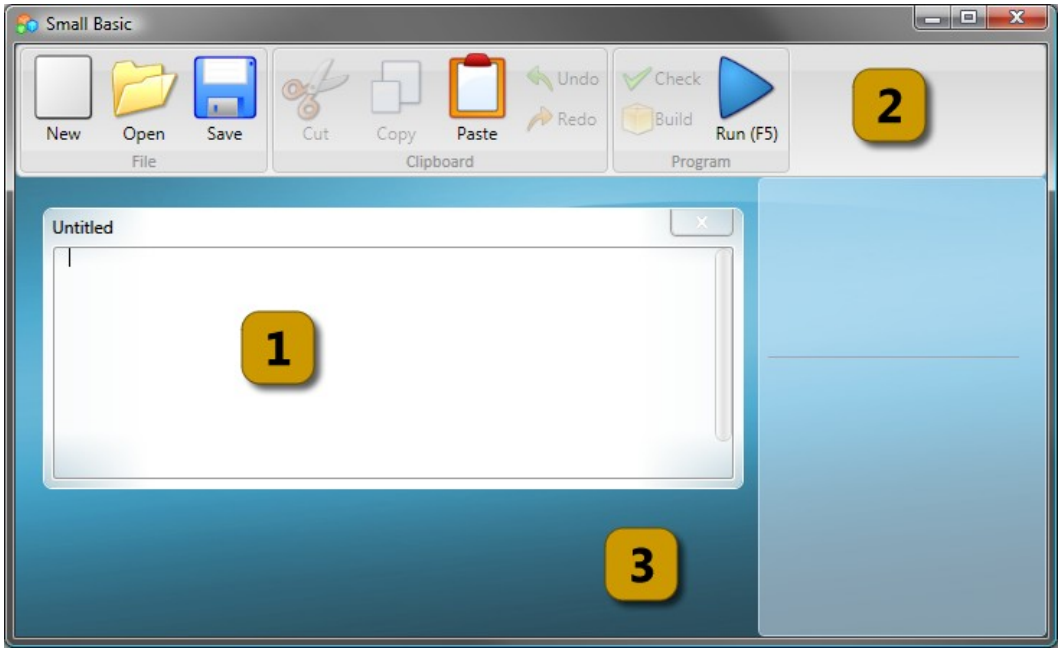
## Small Basic a programování

Programování je proces vytváření počítačového software za použití programovacích jazyků. Stejně jako my mluvíme a rozumíme česky, anglicky nebo německy, počítače rozumí programům napsaným v různých jazycích. Nazývají se programovací jazyky. Na začátku bylo jen málo programovacích jazyků, které bylo snadné se naučit a porozumět jim. Ale jak se počítače a software neustále zdokonalovaly, programovací jazyky se rychle vyvíjely, shromažďovaly stále více různých funkcí. Výsledkem je, že většina programovacích jazyků je pro začátečníky velmi složitá. Tato skutečnost začala odrazovat lidi od toho, aby se začali učit programovat.

Small Basic je programovací jazyk navržený tak, aby udělal programování velmi snadné, dostupné a zábavné pro začátečníky. Záměr jazyka Small Basic je prolomit bariéru a sloužit jako odrazový můstek do úžasného světa programování.

## Prostředí Small Basic

Začněme s rychlým úvodem do prostředí Small Basic. Když poprvé spustíte SmallBasic.exe, uvidíte okno, podobné tomu na následujícím obrázku.




Obrázek 1 - Prostředí Small Basic

Toto je prostředí Small Basic, ve kterém budeme psát a spouštět naše programy. Toto prostředí má několik různých prvků, které jsou identifikovány čísly.

**Editor** [1] je okno, do kterého budeme psát naše programy. Pokud otevřete vzorový nebo dříve uložený program, zobrazí se v tomto editoru. Můžete jej upravit a uložit pro pozdější použití.

Můžete také otevřít více programů a pracovat s nimi najednou. Každý program, se kterým pracujete, bude zobrazen ve svém vlastním editoru. Editor obsahující program, se kterým právě pracujete, se nazývá *aktivní editor*.

**Panel nástrojů** [2] se používá k provádění příkazů pro *aktivní editor* nebo celé prostředí. Postupně se s těmito příkazy naučíme pracovat.

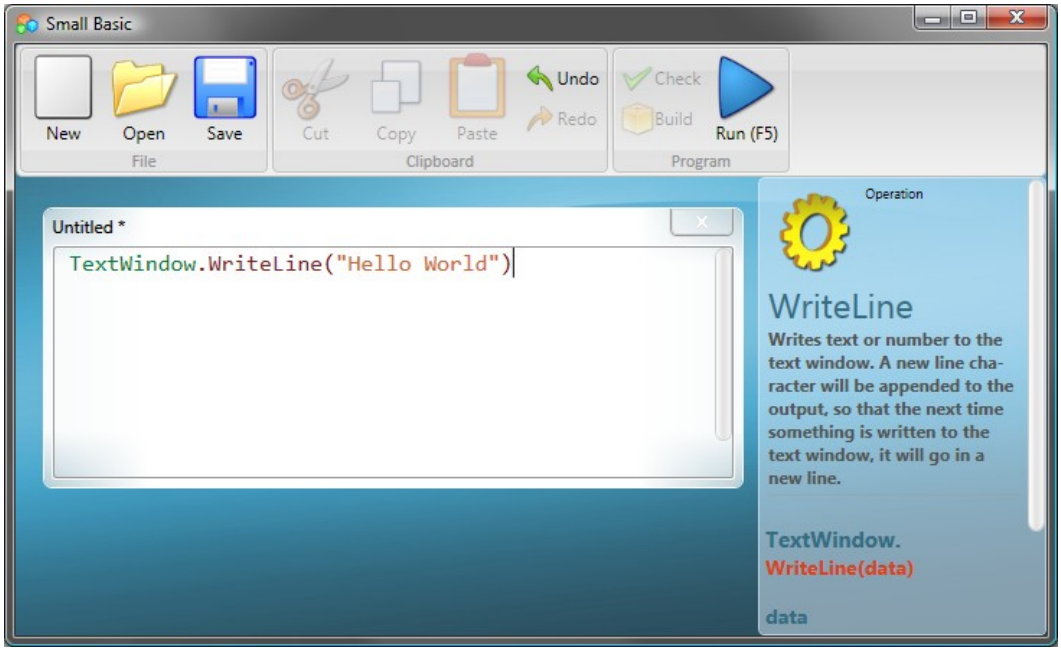
**Povrch** [3] je místo, kde jsou shromážděna všechna okna editorů.

## Náš první program

Teď, když jste se seznámili s prostředím Small Basic, pustíme se do programování. Jak již bylo uvedeno, editor je místo, kam píšeme své programy. Tak si to vyzkoušíme a napíšeme do editoru následující řádek.

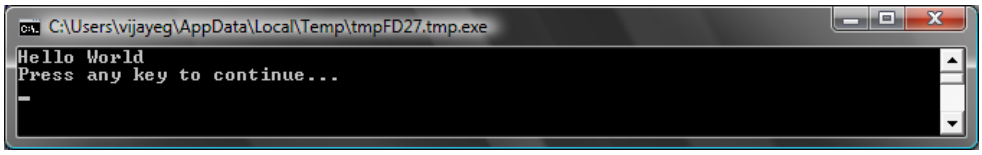
```
TextWindow.WriteLine("Ahoj světe!")
```

Toto je náš první program. Pokud jste to napsali správně, měli byste vidět něco jako na obrázku níže.




**Obrázek 2 – První program**

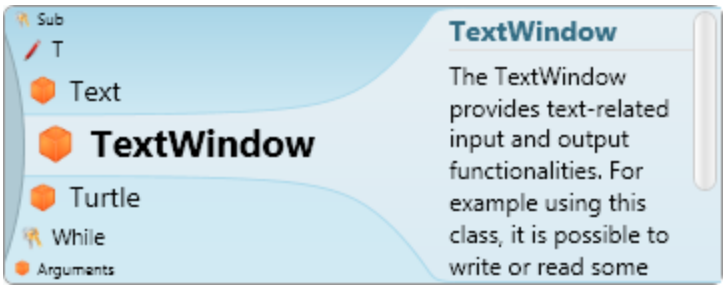
Teď, když jsme napsali náš první program, pojďme jej zkusit spustit, To můžeme udělat buďto kliknutím na tlačítko *Run (spustit)* na panelu nástrojů, nebo stisknutím klávesy F5. Pokud vše bude správně, náš program by měl vypadat tak, jako na obrázku níže.




**Obrázek 3 – Výstup prvního programu**

Gratulujeme! Právě jste napsali a spustili první Small Basic program. Je velmi malý a jednoduchý, ale je to velký krok k tomu stát se skutečným programátorem! Teď ještě jeden detail, než se pustíme do většího programu. Musíme rozumět tomu, co se právě stalo – co jsme přesně počítači řekli a jak věděl, co udělat? V následující kapitole tento program analyzujeme, abychom tomu porozuměli.

*Když jste psali svůj první program, mohli jste si všimnout, že se objevilo vyskakovací okno se seznamem položek (*Obrázek 4*). Tomu se říká „intellisense“ a pomáhá vám psát programy rychleji. Seznam můžete procházet stiskem šipek nahoru a dolů a když najdete to, co potřebujete, stiskněte enter a položka se do vašeho programu vloží.*




Obrázek 4 - Intellisense

## Uložení programu

Pokud chcete zavřít Small Basic a později se vrátit k práci na vašem programu, můžete jej uložit. Je dobré čas od času programy ukládat, abyste o ně v případě náhlého vypnutí nebo výpadku proudu nepřišli. Aktuální program můžete uložit buď kliknutím na tlačítko *Save (uložit)* nebo stisknutím klávesové zkratky Ctrl+S.

*Kapitola 2*

# Porozumění našemu prvnímu programu

## Co je vlastně počítačový program?

Program je soubor instrukcí pro počítač. Tyto instrukce počítači přesně říkají co dělat a počítač je vždycky provádí. Stejně jako lidé, počítače mohou provádět pouze instrukce v jazykách, kterým rozumí. Těm se říká programovací jazyky. Je mnoho jazyků, kterým může počítač rozumět a **Small Basic** je jeden z nich.

Představte si konverzaci mezi vámi a vaším přítelem. Vy a vaši přátelé používáte slova organizovaná do vět, které zprostředkovávají přenos informací. Podobně i programovací jazyky obsahují soubory slov, které mohou být organizovány do vět, které předávají počítači informace. A programy jsou standardně soubory vět (někdy jen několika a někdy mnoha tisíců), které dávají smysl programátorovi stejně jako počítači.

*Je mnoho jazyků, kterým počítač rozumí. Java, C++, Python, VB, atd. jsou všechnomoderní počítačové jazyky používané pro vývoj programů – od jednoduchých po komplexní.*

## Programy Small Basic

Typický Small Basic program se skládá ze spousty *příkazů*. Každý řádek programu reprezentuje příkaz a každý příkaz je instrukce pro počítač. Když požádáme počítač o spuštění Small Basic programu, vezme program a přečte první příkaz. Rozumí tomu, co jsme chtěli říct, a vykoná naši instrukci. Když je první instrukce hotova, vrátí se k programu a přečte a vykoná druhý řádek. Takto pokračuje, dokud nenarazí na konec programu. V ten okamžik program končí.

## Zpět k našemu prvnímu programu

Toto je první program, který jsme napsali:

```
TextWindow.WriteLine("Ahoj světe!")
```

Toto je velmi jednoduchý program, který obsahuje jeden *příkaz*. Tento příkaz řekne počítači, aby vypsal řádek s textem **Ahoj světe!** do textového okna.

V počítačově mysli se to přeloží jako:

```
Napiš Ahoj světe!
```

Možná jste si už všimli, že příkaz se skládá ze tří menších částí, stejně jako věta se skládá ze slov. V prvním příkazu jsme měli 3 různé části:

a) TextWindow
b) WriteLine
c) "Ahoj světe!"

Tečka, závorky a uvozovky jsou všechno znaménka, která se musí umístit na odpovídající pozice v příkazu, aby počítač rozuměl našemu záměru.

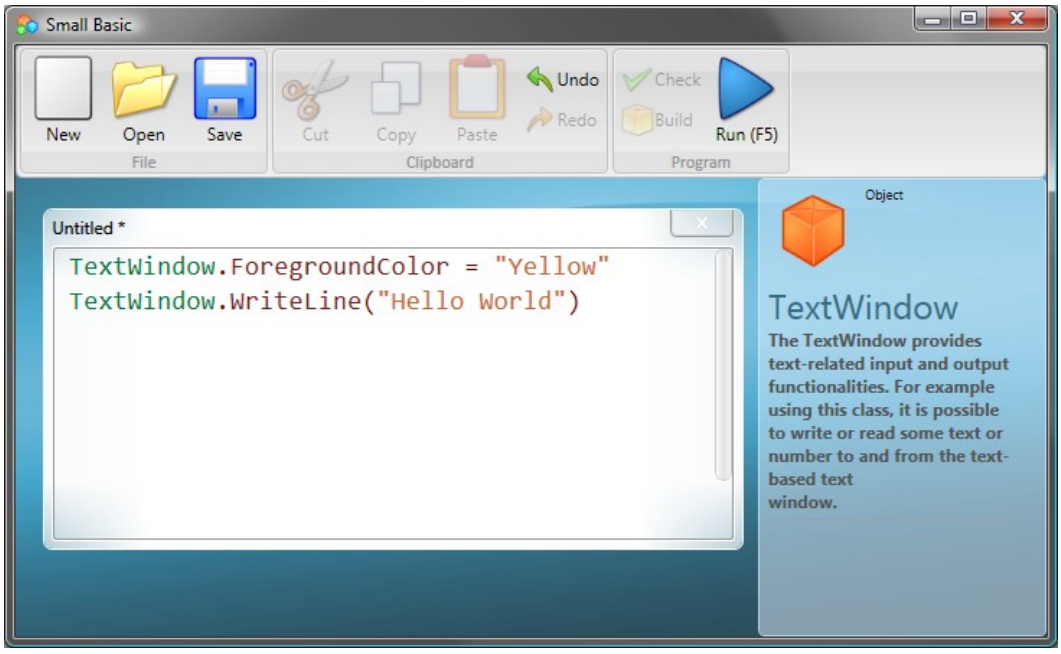
Možná si vzpomínáte na černé okno, které se objevilo, když jsme poprvé spustili náš program. Toto okno se nazývá TextWindow (textové okno), někdy také označováno jako konzole. Tam jde výsledek našeho programu. **TextWindow** v našem programu se nazývá *objekt*. Je spousta takových objektů dostupných pro použití v našich programech. S těmito objekty můžeme provádět několik různých *operací*. Už jsme použili operaci *WriteLine (napsat a odřádkovat)*. Možná jste si také všimli, že operace WriteLine je následována **Ahoj světe!** v uvozovkách Tento text je předán jako vstup operace WriteLine, která jej poté vypíše uživateli. Toto se nazývá *vstup* operace. Některé operace mají více vstupů, některé naopak žádné.

*Znaménka jako uvozovky, mezery a závorky jsou v počítačovém programu velmi důležité. Jejich umístění a počet může změnit smysl toho, co je napsáno.*

## Náš druhý program

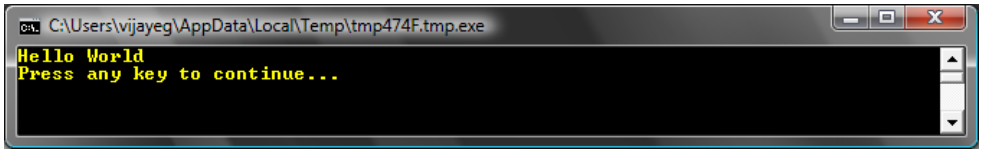
Teď jste porozuměli našemu prvnímu programu, pojďme dál a zkusme jej vylepšit přidáním nějakých barev.

```
TextWindow.ForegroundColor = "Yellow"
TextWindow.WriteLine("Ahoj světe!")
```




Obrázek 5 – Přidání barev

Když spustíte program výše, zjistíte, že vypíše stejnou frázi „Ahoj světe!“ do textového okna, ale vypíše ji žlutě místo šedě jako dřív.




Obrázek 6 – Žlutá barva

Všimněte si nového příkazu, který jsme přidali do původního programu. Používá nové slovo, *ForegroundColor (barva popředí)*, za kterým je znak rovná se a hodnota *„Yellow“ (žlutá)*. To znamená, že jsme *přiřadili* „Yellow“ do *ForegroundColor*. Rozdíl mezi ForegroundColor a operací WriteLine je, že ForegroundColor nemá žádné vstupy ani nepotřebuje žádné závorky. Místo toho bylo následováno znakem *rovná se* a slovem. ForegroundColor definujeme jako *vlastnost* objektu TextWindow. Zde je seznam hodnot, které je možné přiřadit vlastnosti ForegroundColor. Zkuste nahradit „Yellow“ některou z následujících (bez slov v závorkách, to jsou pouze překlady) a podívejte se, co se stane – nezapomeňte na uvozovky, ty jsou zde vyžadovány.

```
Black (černá)
Blue (modrá)
Cyan (azurová)
Gray (šedá)
Green (zelená)
Magenta (růžová)
Red (červená)
White (bílá)
Yellow (žlutá)
DarkBlue
DarkCyan
DarkGray
DarkGreen
DarkMagenta
DarkRed
DarkYellow
```

Kapitola 3

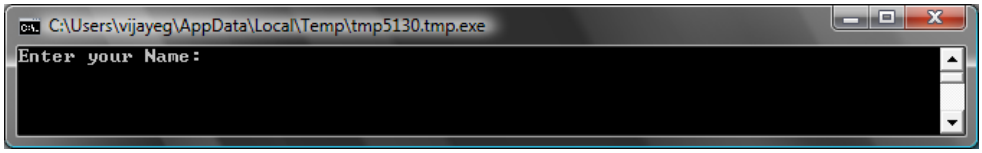
# Úvod do proměnných

## Použití proměnných v našem programu

Nebylo by hezké, kdyby náš program řekl „Ahoj “ se jménem uživatele, místo obecného „Ahoj světe!“? K tomu se nejdřív musíme uživatele zeptat na jméno, někde jej uložit a pak vypsat „Ahoj“ s jeho jménem. Ukažme si, jak to můžeme udělat:

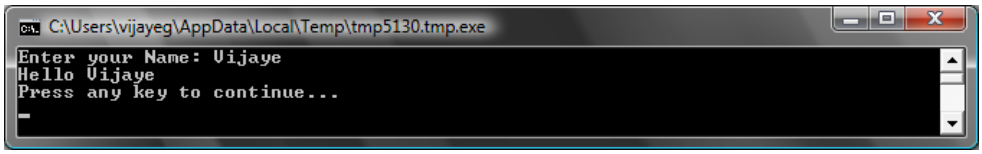
```
TextWindow.Write("Napište své jméno: ")
jmeno = TextWindow.Read()
TextWindow.WriteLine("Ahoj " + jmeno)
```

Když napíšete a spustíte tento program, uvidíte něco jako:




Obrázek 7 – Dotaz na jméno uživatele

A když napíšete své jméno a stisknete enter, uvidíte následující výstup:




Obrázek 8 - Pozdrav

Teď když spustíte program znovu, budete znovu dotázáni na tu samou otázku. Můžete napsat jiné jméno a počítač vypíše Ahoj s tímto jménem (bohužel nebude skloňované, to nejde).

## Analýza programu

V programu, který jste právě spustili, mohl následující řádek upoutat vaši pozornost:

```
jmeno = TextWindow.Read()
```

*Read() (načíst)* vypadá podobně jako *WriteLine() (vypsat a odřádkovat)*, ale bez vstupů. Je to operace, která říká počítači, aby počkal, až uživatel něco napíše a stiskne enter. Jak jednou stiskne enter, vezme to, co uživatel napsal a vrátí se k programu. Cokoliv uživatel napíše, se uloží do *proměnné* nazvané **jmeno**. *Proměnná* je místo, kam můžete dočasně ukládat hodnoty a použít je později. V řádku výše bylo **jmeno** použito pro uložení jména uživatele.

Další řádek je také zajímavý:

```
TextWindow.WriteLine("Ahoj " + jmeno)
```

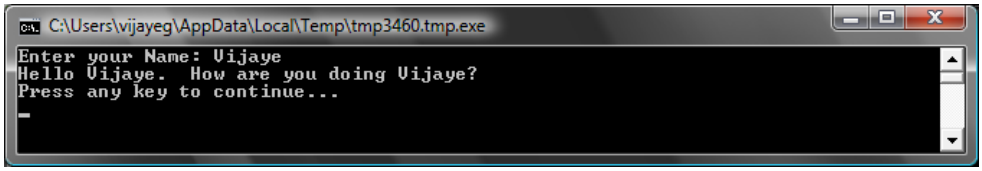
Zde použijeme hodnotu uloženou v naší proměnné, **jmeno**. Vezmeme hodnotu ve **jmeno**, přidáme ji za „Ahoj“ a vypíšeme do textového okna.

Jak je jednou proměnná nastavena, můžete ji znovu použít, kolikrát chcete. Například můžete udělat následující:

Write, *stejně jako* WriteLine, *je jiná operace pro textové okno. Write (vypsat) vám umožní něco napsat, ale neodřádkuje se - následující text bude na stejném řádku jako ten aktuální.*

```
TextWindow.Write("Napište své jméno: ")
jmeno = TextWindow.Read()
TextWindow.Write("Ahoj " + jmeno + "!  ")
TextWindow.WriteLine("Jak to jde, " + jmeno + "?")
```

A uvidíte následující výstup:



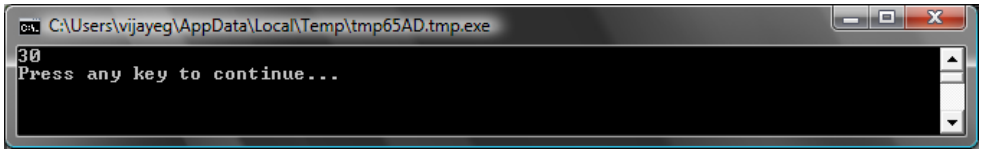

Obrázek 9 – Opakované použití proměnné

## Hraní si s čísly

Právě jsme viděli, jak můžete použít proměnné k uložení jména uživatele. V následujících několika programech uvidíme, jak můžeme ukládat a manipulovat s čísly v proměnných. Začněme s velmi jednoduchým programem:

```
cislo1 = 10
cislo2 = 20
cislo3 = cislo1 + cislo2
TextWindow.WriteLine(cislo3)
```

Když tento program spustíte, uvidíte následující výstup:




Obrázek 10 – Sčítání dvou čísel

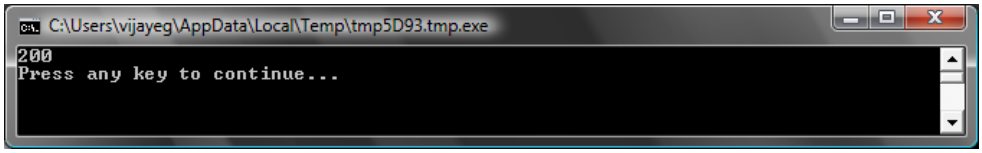
Na prvním řádku programu přiřazujete proměnné **cislo1** hodnotu 10. A ve druhém řádku přiřazujete proměnné **cislo2** hodnotu 20. Ve třetím řádku sčítáte **cislo1** a **cislo2** a výsledek ukládáte do **cislo3**. Takže v tomto případě bude mít **cislo3** hodnotu 30. A to se vypsalo do textového okna.

*Všimněte si, že čísla kolem sebe nemají uvozovky. Pro čísla nejsou uvozovky nutné. Potřebujete je pouze pokud chcete použít text.*

Teď zkusme program trochu modifikovat a uvidíme výsledek:

```
cislo1 = 10
cislo2 = 20
cislo3 = cislo1 * cislo2
TextWindow.WriteLine(cislo3)
```

Program vynásobí **cislo1** a **cislo2** a výsledek uloží do **cislo3**. Toto bude výsledek:




Obrázek 11 – Násobení dvou čísel

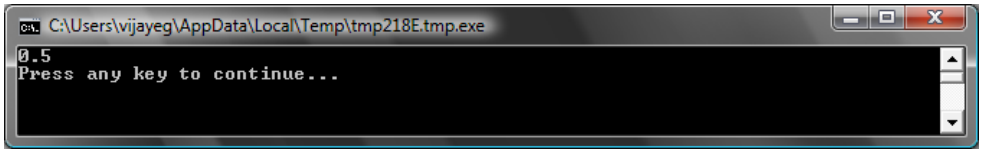
Podobně můžete odečítat nebo dělit. Zde je odečítání:

```
cislo3 = cislo1 - cislo2
```

Symbol pro dělení je lomítko „/“. Program bude vypadat takto.:

```
cislo3 = cislo1 / cislo2
```

A výsledek dělení bude:




Obrázek 12 – Dělení dvou čísel

## Jednoduchý převaděč teplot

V následujícím programu použijeme vzorec $℃ = \frac{5(°F-32)}{9}$ k převodu ze stupňů Fahrenheita na stupně Celsia.

Nejdříve dostaneme od uživatele teplotu v °F a uložíme ji do proměnné. Existuje speciální operace, která nám umožní načítat čísla od uživatele - **TextWindow.ReadNumber (načíst číslo)**.

```
TextWindow.Write("Zadejte teplotu v °F: ")
fahr = TextWindow.ReadNumber()
```

Když máme teplotu ve stupních Fahrenheita uloženou v proměnné, můžeme ji takto převést do stupňů Celsia:

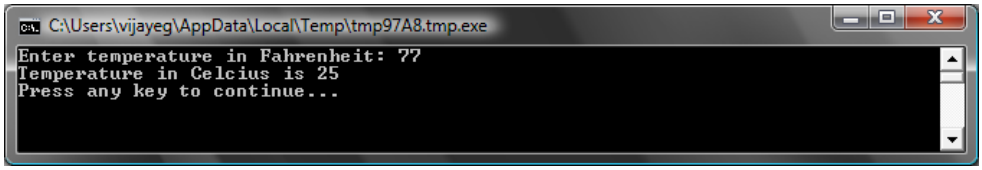
```
celsius = 5 * (fahr - 32) / 9
```

Závorky řeknou počítači, aby nejdříve spočítal **fahr – 32** a poté zbytek. Vše, to potom musíme udělat, je vypsat výsledek uživateli. Výsledkem bude tento program:

```
TextWindow.Write("Zadejte teplotu v °F: ")
fahr = TextWindow.ReadNumber()
```

```
celsius = 5 * (fahr - 32) / 9
TextWindow.WriteLine("Teplota v °C je " + celsius)
```

A výsledkem bude:




Obrázek 13 – Převod teploty

*Kapitola 4*

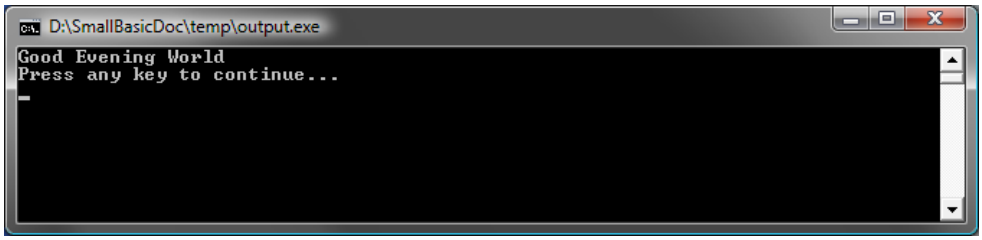
# Podmínky a větvení

Zpět k prvnímu programu. Nebylo by skvělé místo obecného *Ahoj světe* říct *Dobré ráno světe* nebo *Dobrý večer světe* v závislosti na denní době? Náš příští program vypíše *Dobré ráno světe*, pokud je méně než 12 hodin a *Dobrý večer světe* pokud je víc než 12 hodin.

```
If (Clock.Hour < 12) Then
  TextWindow.WriteLine("Dobré ráno světe")
EndIf
If (Clock.Hour >= 12) Then
  TextWindow.WriteLine("Dobrý večer světe")
EndIf
```

V závislosti na tom, kdy program spustíte, uvidíte následující výstup:


**Obrázek 14 – Dobré ráno světe**




Obrázek 15 – Dobrý večer světe

Analyzujme první tři řádky programu. Už jste jistě zjistili, že říkají počítači, aby vypsal „Dobré ráno světe“, pokud je Clock.Hour menší než 12. Slova **If (když)**, **Then (potom)** a **Endif (konec když)** jsou speciální slova, kterým počítač rozumí. Slovo **If** je vždy následováno podmínkou, která je v tomto případě (**Clock.Hour < 12**). Pamatujte si, že závorky jsou nutné, aby počítač rozuměl vašim záměrům. Podmínka je následována slovem **Then** a operací k vykonání. Na konci operace je vždy **EndIf**. Toto říká počítači, že podmíněná konstrukce je u konce.

*V jazyce Small Basic můžete použít objekt* Clock (hodiny) *pro přístup k aktuálnímu datu a času. Obsahuje vlastnosti jako Day (den), Month (měsíc), Year (rok), Hour (hodina), Minute (minuta) Second (sekunda), atd.*

Mezi **Then** a **EndIf** může být víc než jedna operace a počítač je všechny vykoná, pokud podmínka platí. Například můžete napsat něco jako toto:

```
If (Clock.Hour < 12) Then
  TextWindow.Write("Dobré ráno. ")
  TextWindow.WriteLine("Jaká byla snídaně?")
EndIf
```

## Nebo

V programu na začátku kapitoly jste si mohli všimnout, že druhá podmínka je poněkud zbytečná. Hodnota **Clock.Hour** může být menší než 12 nebo ne. Nemuseli jsme dělat druhé ověřování. V takovýchto případech můžeme zkrátit dva příkazy **if..then..endif** do jednoho za použití nového slova, **Else (nebo)**.

Kdybychom měli přepsat program, aby používal **Else**, vypadal by takto:

```
If (Clock.Hour < 12) Then
  TextWindow.WriteLine("Dobré ráno světe")
Else
  TextWindow.WriteLine("Dobrý večer světe")
```

```
EndIf
```

Tento program udělá úplně to samé jako ten předchozí, což nám přináší velmi důležitou lekci v programování:

> *V programování je obvykle mnoho způsobů jak udělat stejnou věc. Někdy dává jeden způsob větší smysl než ten druhý. Volba je na programátorovi. Jak budete psát více programů a získávat více zkušeností, začnete si všímat rozdílných technik a jejich výhod a nevýhod.*

## Odsazování

Ve všech příkladech můžete vidět, jak jsou příkazy mezi *If*, *Else* a *EndIf* odsazeny. Toto odsazování není nutné. Počítač bez nich programu snadno porozumí. Nicméně, pomáhají nám se lépe orientovat ve struktuře programu. Proto je většinou bráno jako dobrá praxe odsazovat příkazy mezi takovými bloky.

## Sudé nebo liché

Teď, když umíme používat strukturu **If..Then..Else..EndIf**, napišme program, který pozná, zda je zadané číslo sudé nebo liché.

```
TextWindow.Write("Zadejte číslo: ")
cis = TextWindow.ReadNumber()
zbytek = Math.Remainder(cis, 2)
If (zbytek = 0) Then
  TextWindow.WriteLine("Číslo je sudé")
Else
  TextWindow.WriteLine("Číslo je liché")
EndIf
```

Když spustíte program, uvidíte podobný výstup:

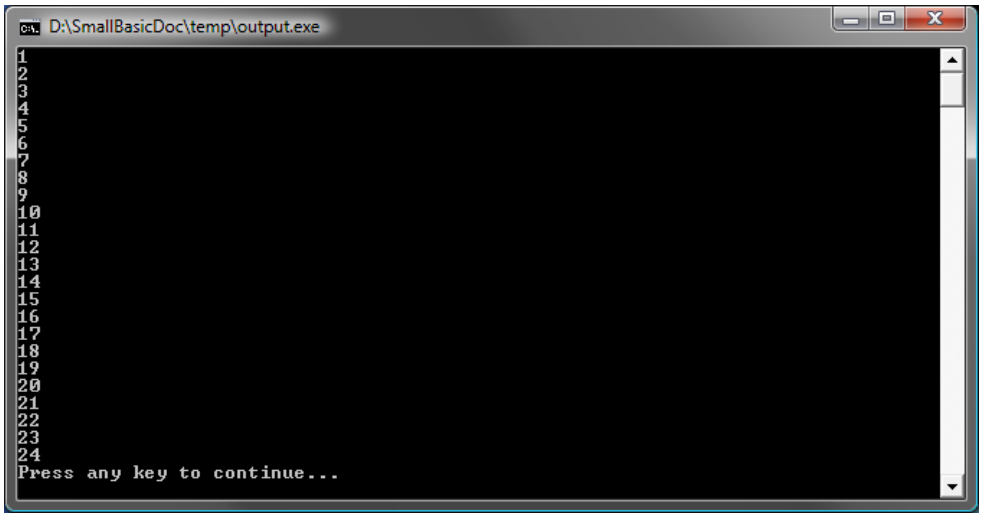

Obrázek 16 – Sudé nebo liché

V tomto programu jsme se setkali s novou užitečnou operací, **Math.Remainder (zbytek)**. Jak jste si možná již všimli, **Math.Remainder** vydělí první číslo druhým a vrátí zbytek.

## Větvení

Ve druhé kapitole jste se naučili, že počítač provádí jeden příkaz v jeden okamžik, vždy odshora dolů. Ale je tu speciální příkaz, který umožní skočit před jiný příkaz mimo pořadí. Podívejme se na následující program:

```
i = 1
start:
TextWindow.WriteLine(i)
i = i + 1
If (i < 25) Then
  Goto start
EndIf
```




Obrázek 17 - Používání Goto

V programu výše jsme přiřadili hodnotu 1 do proměnné **i**. Poté jsme přidali nový příkaz, který končí dvojtečkou (:)

```
start:
```

Tomu se říká *návěští*. Návěští jsou jako záložky, kterým počítač rozumí. Můžete je pojmenovat, jak chcete a můžete použít tolik návěští, kolik chcete, jen musí být každé pojmenované jinak.

Další zajímavý příkaz je tento:

```
i = i + 1
```

¨Toto řekne počítači, aby přidal 1 k proměnné **i** a přiřadil výsledek zpět do **i**. Takže pokud hodnota **i** byla na začátku 1, po tomto řádku bude 2.

A nakonec,

```
If (i < 25) Then
  Goto start
EndIf
```

Toto je část, která říká počítači, že pokud hodnota **i** je menší než 25, má začít spouštět příkazy od návěští **start**.

## Nekonečné spouštění

Za použití příkazu **Goto** můžete nechat počítač něco opakovat, kolikrát chcete. Například můžete vzít program zjišťující sudé nebo liché číslo, upravit ho jako níže a program poběží navždy. Můžete jej zastavit kliknutím na tlačítko Zavřít (X) v pravém horním rohu okna.

```
zacatek:
TextWindow.Write("Zadejte číslo: ")
cis = TextWindow.ReadNumber()
zbytek = Math.Remainder(cis, 2)
If (zbytek = 0) Then
  TextWindow.WriteLine("Číslo je sudé")
Else
  TextWindow.WriteLine("Číslo je liché")
EndIf
Goto zacatek
```


Obrázek 18 – Sudé nebo liché běžící navždy

*Kapitola 5*

# Cykly

## For cyklus

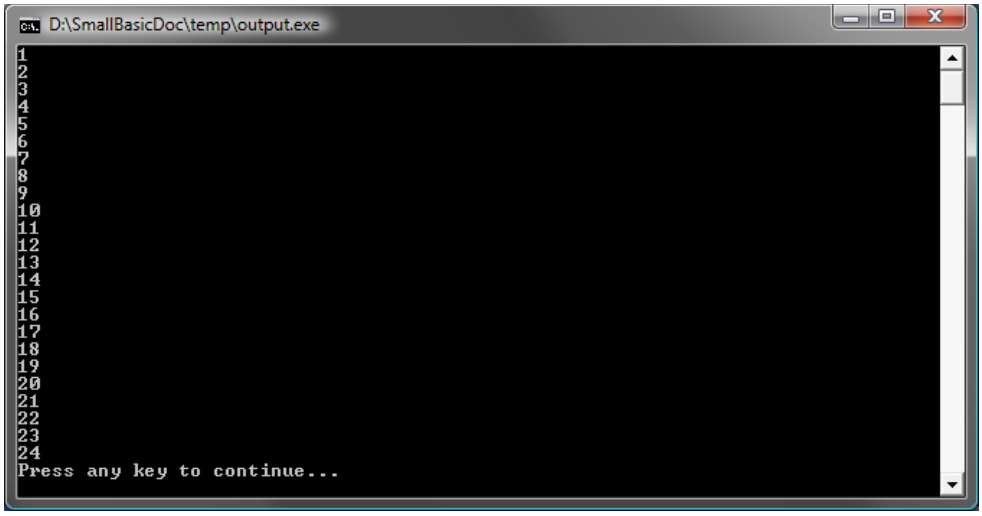
Vezměme si program, který jsme napsali v předchozí kapitole.

```
i = 1
start:
TextWindow.WriteLine(i)
i = i + 1
If (i < 25) Then
  Goto start
EndIf
```

Tento program vypisuje čísla od 1 do 24 popořadě. Tento proces zvyšování proměnné je v programování tak běžný, že programovací jazyky většinou poskytují snazší metodu, jak toto řešit. Program nahoře je stejný jako program dole:

```
For i = 1 To 24
  TextWindow.WriteLine(i)
EndFor
```

A výsledek je:



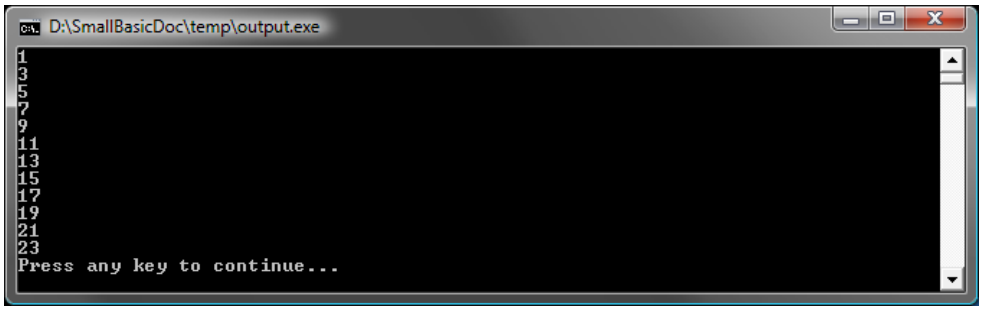

Obrázek 19 – Používání For cyklu

Všimněte si, že jsme zmenšili 7řádkový program na 3řádkový a stále dělá to samé! Pamatujete, jak jsme říkali, že je většinou několik způsobů jak udělat tu samou věc? Toto je skvělý příklad.

**For..EndFor** se nazývá *cyklus*. Umožní vám vzít proměnnou, dát jí počáteční a konečnou hodnotu a nechat počítač zvyšovat proměnnou za vás. Pokaždé když počítač hodnotu zvýší, spustí příkazy mezi **For** a **EndFor**.

Ale pokud byste chtěli, aby se proměnná zvyšovala o 2 namísto o 1 – jako byste řekli, že chcete vypsat všechna lichá čísla mezi 1 a 24 – můžete k tomu využít také tento cyklus.

```
For i = 1 To 24 Step 2
  TextWindow.WriteLine(i)
EndFor
```




Obrázek 20 – Pouze lichá čísla

**Step 2** říká počítači, že má zvyšovat hodnotu **i** o 2 namísto 1. Použitím **Step** můžete určit jakýkoliv přírůstek chcete. Dokonce můžete zadat zápornou hodnotu a počítač bude počítat pozpátku, jako v příkladu níže:

```
For i = 10 To 1 Step -1
  TextWindow.WriteLine(i)
EndFor
```


Obrázek 21 – Počítání pozpátku

## While cyklus

While cyklus je další metoda, která je užitečná hlavně pokud není znám počet opakování. Zatímco For cyklus běží po určený počet kroků, While cyklus běží, dokud daná podmínka platí. V příkladu níže dělíme číslo dvěma, dokud je výsledek větší než 1.

```
cislo = 100
While (cislo > 1)
  TextWindow.WriteLine(cislo)
  cislo = cislo / 2
EndWhile
```


Obrázek 22 – Dělení čísla dvěma

V programu výše jsme přiřadili hodnotu 100 do proměnné *cislo* a while cyklus běží, dokud je toto číslo větší než 1. Uvnitř cyklu vypisujeme číslo a poté jej dělíme dvěma. Jak jsme očekávali, výsledkem tohoto programu jsou čísla, která jsou zmenšována o polovinu.

Bylo by velmi těžké napsat tento program s použitím For cyklu, protože nevíme, kolikrát bude cyklus běžet. S while cyklem je snadné zkontrolovat podmínku a požádat počítač, aby buď v cyklu pokračoval, nebo skončil.

Bude zajímavé všimnout si, že každý while cyklus může být rozbalen do příkazů If..Then. Například program nahoře může být přepsán na program dole s naprosto stejným výsledkem.

```
cislo = 100
start:
TextWindow.WriteLine(cislo)
cislo = cislo / 2

If (cislo > 1) Then
  Goto start
EndIf
```

*Ve skutečnosti počítač vnitřně přepisuje každý While cyklus do příkazů, které používají If..Then s jedním nebo více Goto příkazy.*

*Kapitola 6*

# Práce s grafikou

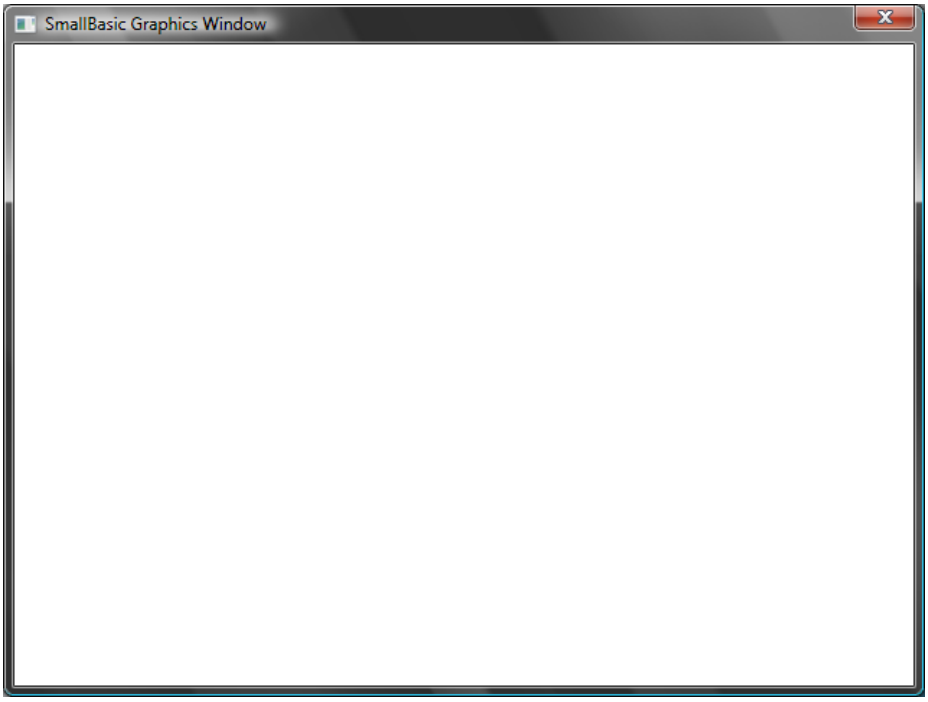
Ve všech čtyřech příkladech jsme používali textové okno, abychom vysvětlili prvky jazyka Small Basic. Nicméně, Small Basic disponuje sadou grafických prvků, které začneme v této kapitole probírat.

## Představení GraphicsWindow

Stejně jako nám textové okno umožnilo pracovat s textem a čísly, Small Basic také poskytuje **GraphicsWindow (grafické okno)**, které můžeme použít ke kreslení věcí. Začněme se zobrazením grafického okna.

```
GraphicsWindow.Show()
```

Když spustíte tento program, zjistíte, že namísto obvyklého černého textového okna získáte bílé okno jako to níže zobrazené. Teď se s tímto oknem nedá moc dělat. Ale bude to to hlavní okno, se kterým budeme v této kapitole pracovat. Můžete jej zavřít kliknutím na tlačítko „X“ v pravém horním rohu okna.



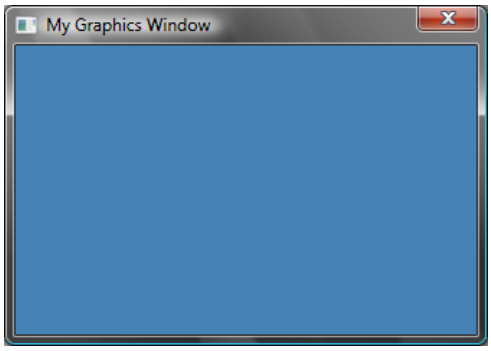

Obrázek 23 – prázdné grafické okno

## Nastavení grafického okna

Grafické okno vám umožní přizpůsobit svůj vzhled vašim požadavkům. Můžete změnit nadpis, pozadí a velikost. Pojďme dál a změňte to, abychom se s tímto oknem seznámili.

```
GraphicsWindow.BackgroundColor = "SteelBlue"
GraphicsWindow.Title = "Mé grafické okno"
GraphicsWindow.Width = 320
GraphicsWindow.Height = 200
GraphicsWindow.Show()
```

Zde je obrázek toho, jak přizpůsobené grafické okno vypadá. Můžete změnit barvu pozadí na jednu z mnoha variant vypsaných v Příloze B. Pohrajte si s těmito vlastnostmi, abyste viděli, jak se změní vzhled okna.

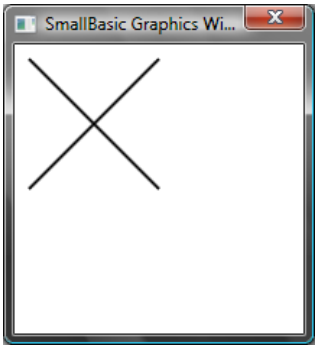



Obrázek 24 – Vlastní grafické okno

## Kreslení čar

Jak jednou máme grafické okno, můžeme na ně kreslit tvary, text a dokonce obrázky. Začněme s kreslením nějakých jednoduchých tvarů. Zde je program, který nakreslí několik čar do grafického okna.

```
GraphicsWindow.Width = 200
GraphicsWindow.Height = 200
GraphicsWindow.DrawLine(10, 10, 100, 100)
GraphicsWindow.DrawLine(10, 100, 100, 10)
```




Obrázek 25 – Křížek

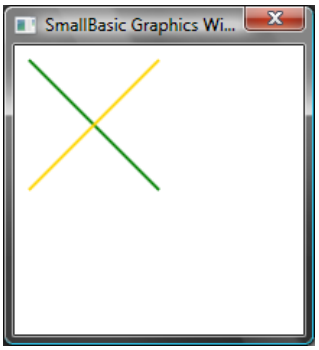
První dva řádky programu nastavují okno a další dva vykreslí křížek. První dvě čísla, která následují *DrawLine (vykreslit čáru)*, určují počáteční x a y souřadnici a další dvě určují konečnou x a y souřadnici. Zajímavé je, že

*Místo používání názvů barev můžete použít webový kód barvy (#ČČZZMM). Například, #FF0000 určuje červenou, #FFFF00 žlutou a tak.*

počátek souřadného systému (0,0) je v levém horním rohu okna.

Pokud se vrátíme zpět do programu, je zajímavé poznamenat, že Small Basic vám umožňuje upravovat vlastnosti čáry, jako barvu nebo tloušťku. Nejprve upravme barvu čar jako v programu níže.

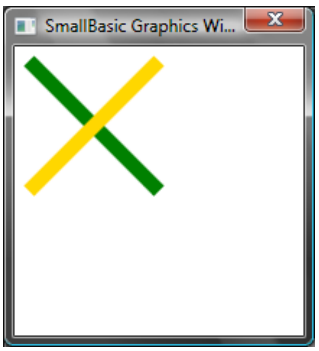
```
GraphicsWindow.Width = 200
GraphicsWindow.Height = 200
GraphicsWindow.PenColor = "Green"
GraphicsWindow.DrawLine(10, 10, 100, 100)
GraphicsWindow.PenColor = "Gold"
GraphicsWindow.DrawLine(10, 100, 100, 10)
```




Obrázek 26 – Změna barvy čáry

Teď změňme i tloušťku. V programu níže ji změníme na 10 místo výchozí 1.

```
GraphicsWindow.Width = 200
GraphicsWindow.Height = 200
GraphicsWindow.PenWidth = 10
GraphicsWindow.PenColor = "Green"
GraphicsWindow.DrawLine(10, 10, 100, 100)
GraphicsWindow.PenColor = "Gold"
GraphicsWindow.DrawLine(10, 100, 100, 10)
```

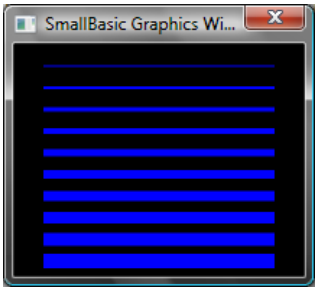



Obrázek 27 – Tlusté barevné čáry

*PenWidth (šířka pera)* a *PenColor (barva pera)* mění pero, se kterým jsou čáry kresleny. To ovlivní nejen čáry, ale i jakýkoliv tvar nakreslený poté, co jsou tyto vlastnosti změněny.

Použitím cyklů, které jsme se v předchozí kapitole naučili, můžeme snadno napsat program, který nakreslí několik čar s rostoucí šířkou pera.

```
GraphicsWindow.BackgroundColor = "Black"
GraphicsWindow.Width = 200
GraphicsWindow.Height = 160
GraphicsWindow.PenColor = "Blue"

For i = 1 To 10
  GraphicsWindow.PenWidth = i
  GraphicsWindow.DrawLine(20, i * 15, 180, i * 15)
endfor
```



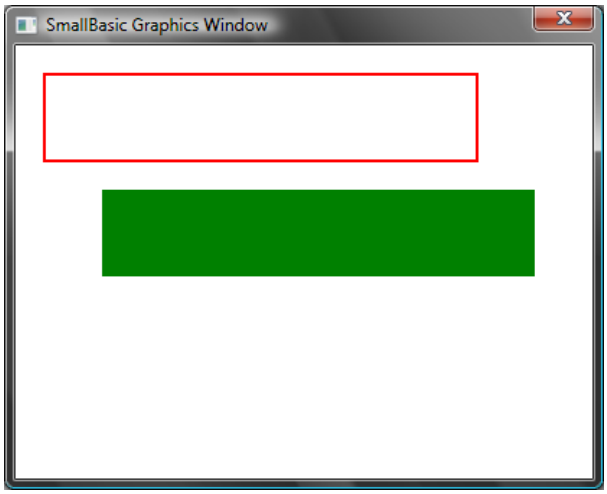

Obrázek 28 – Různé šířky pera

Zajímavá část tohoto programu je cyklus, ve kterém zvyšujeme *PenWidth* pokaždé, když je cyklus spuštěn a poté nakreslíme novou čáru pod tou starou.

## Kreslení a vyplňování tvarů

Při kreslení tvarů jsou většinou dva typy operací pro každý tvar. Jsou to operace *Draw (nakreslit)* a *Fill (vyplnit)*. Kreslící operace nakreslí okraj tvaru a vyplnění jej vybarví štětcem. Na příkladu níže jsou dva obdélníky, z nichž jeden je nakreslen červeným perem a druhý je vyplněn zeleným štětcem.

```
GraphicsWindow.Width = 400
GraphicsWindow.Height = 300

GraphicsWindow.PenColor = "Red"
GraphicsWindow.DrawRectangle(20, 20, 300, 60)

GraphicsWindow.BrushColor = "Green"
GraphicsWindow.FillRectangle(60, 100, 300, 60)
```

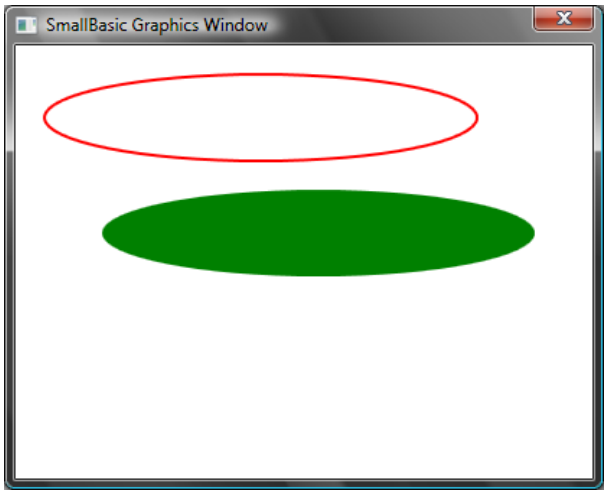



Obrázek 29 Kreslení a vyplnění

Abyste nakreslili nebo vyplnili obdélník, potřebujete čtyři čísla. První dvě reprezentují x a y souřadnice pro levý horní roh obdélníku. Třetí číslo určuje šířku a čtvrté výšku. Ve skutečnosti, to samé platí pro kreslení a vyplňování elips, jak je vidno v programu níže.

```
GraphicsWindow.Width = 400
GraphicsWindow.Height = 300

GraphicsWindow.PenColor = "Red"
GraphicsWindow.DrawEllipse(20, 20, 300, 60)

GraphicsWindow.BrushColor = "Green"
```

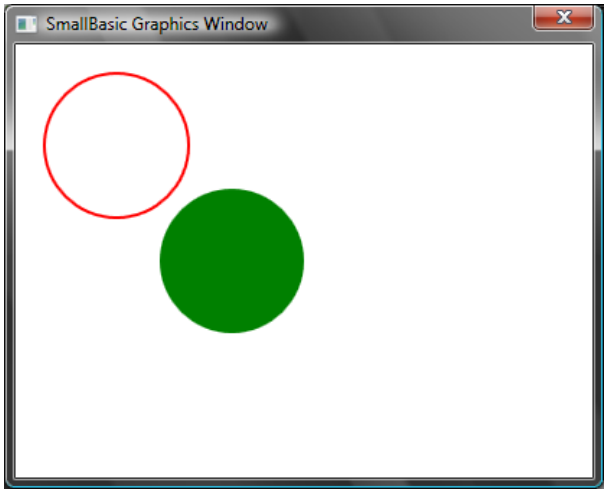
```
GraphicsWindow.FillEllipse(60, 100, 300, 60)
```




Obrázek 30 – Kreslení a vyplňování elips

Elipsy jsou pouze obecné případy kružnic. Pokud chcete kreslit kružnice, musíte zadat stejnou výšku a šířku.

```
GraphicsWindow.Width = 400
GraphicsWindow.Height = 300

GraphicsWindow.PenColor = "Red"
GraphicsWindow.DrawEllipse(20, 20, 100, 100)

GraphicsWindow.BrushColor = "Green"
GraphicsWindow.FillEllipse(100, 100, 100, 100)
```




Obrázek 31 – Kružnice

*Kapitola 7*

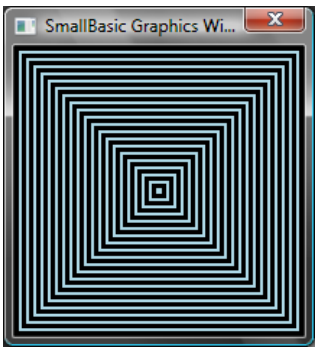
# Zábava s tvary

Teď se trochu pobavíme s tím, co jsme se již naučili. Tato kapitola obsahuje příklady, které ukazují některé zajímavé způsoby, jak kombinovat vše, co jste se již naučili, abyste vytvořili některé dobré programy.

## Pyramida

Zde nakreslíme několik obdélníků s rostoucí velikostí.

```
GraphicsWindow.BackgroundColor = "Black"
GraphicsWindow.PenColor = "LightBlue"
GraphicsWindow.Width = 200
GraphicsWindow.Height = 200

For i = 1 To 100 Step 5
  GraphicsWindow.DrawRectangle(100 - i, 100 - i, i * 2, i *
2)
EndFor
```

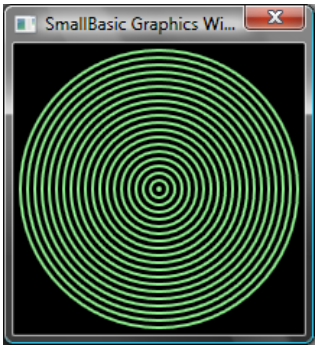



Obrázek 32 - Pyramida

## Soustředné kružnice

Varianta předchozího programu, kreslí kružnice namísto obdélníků.

```
GraphicsWindow.BackgroundColor = "Black"
GraphicsWindow.PenColor = "LightGreen"
GraphicsWindow.Width = 200
GraphicsWindow.Height = 200

For i = 1 To 100 Step 5
  GraphicsWindow.DrawEllipse(100 - i, 100 - i, i * 2, i *
2)
EndFor
```

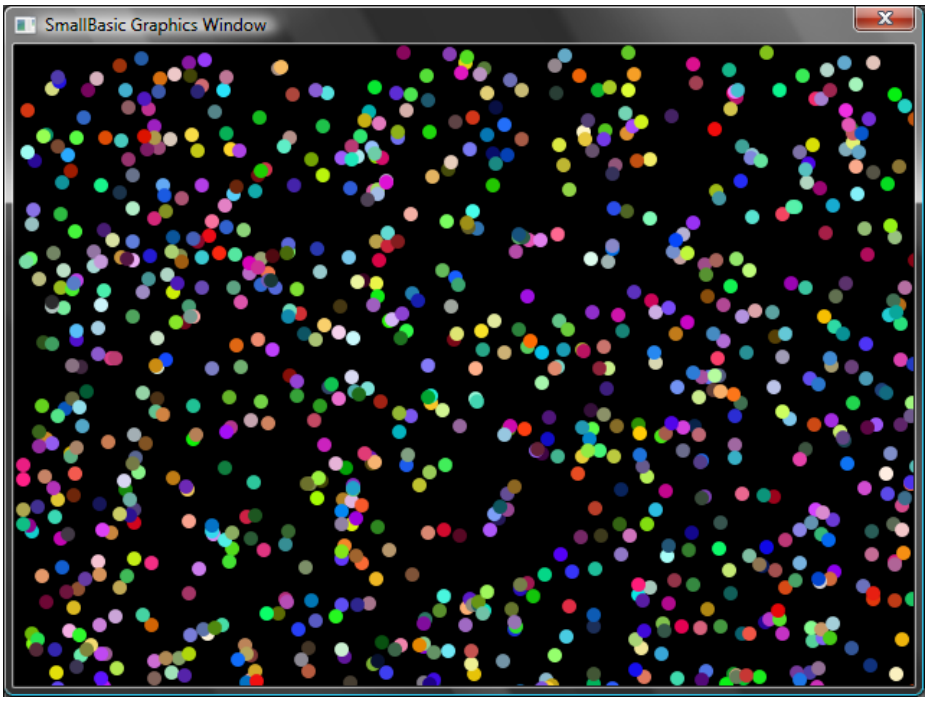



Obrázek 33 – Soustředné kružnice

## Náhoda

Tento program použije operaci GraphicsWindow.*GetRandomColor (získej náhodnou barvu)*, aby nastavil náhodné barvy štětce, a poté použije Math.*GetRandomNumber (získej náhodné číslo)*, aby nastavil x a y souřadnici pro kruhy. Tyto dvě operace mohou být zajímavými způsoby kombinovány k vytváření zajímavých programů, které při každém spuštění dají jiný výsledek.

```
GraphicsWindow.BackgroundColor = "Black"
For i = 1 To 1000
  GraphicsWindow.BrushColor =
GraphicsWindow.GetRandomColor()
  x = Math.GetRandomNumber(640)
  y = Math.GetRandomNumber(480)
  GraphicsWindow.FillEllipse(x, y, 10, 10)
EndFor
```



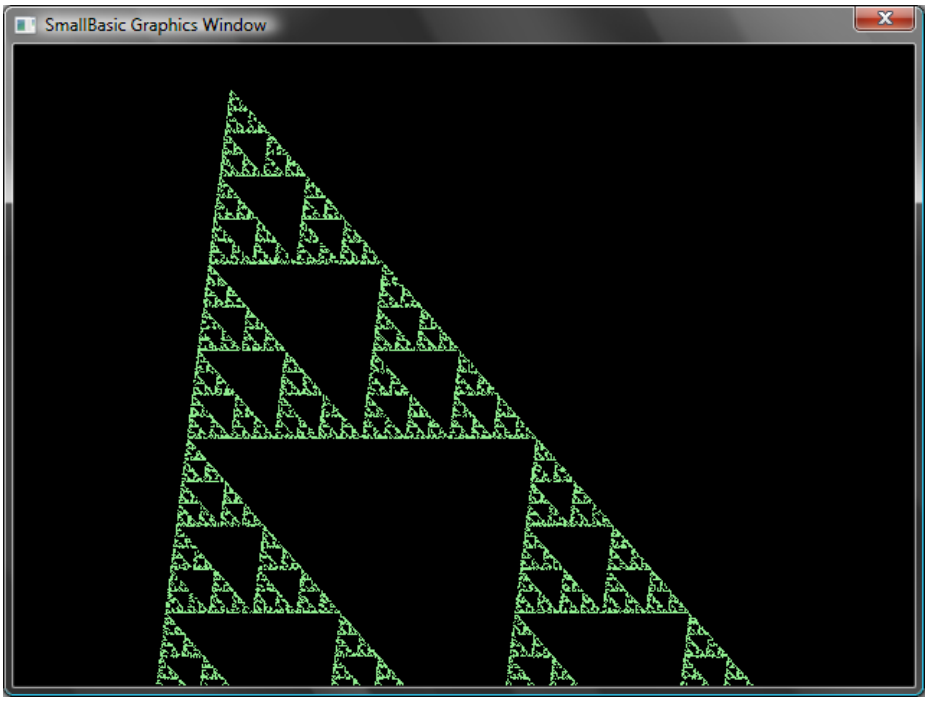

Obrázek 34 – Náhoda

## Fraktály

Následující program nakreslí jednoduchý trojúhelníkový fraktál za použití náhodných čísel. Fraktál je geometrický tvar, který může být rozdělen na části, z nichž každá se přesně podobá původnímu tvaru. V tomto případě, tento program nakreslí stovky trojúhelníků, z nichž každý se bude podobat výchozímu trojúhelníku. A až program po několik vteřin poběží, můžete vidět, jak se trojúhelníky pomalu skládají z teček. Samotná logika je na vysvětlení dost složitá.

```
GraphicsWindow.BackgroundColor = "Black"
x = 100
y = 100
```

```
For i = 1 To 100000
  r = Math.GetRandomNumber(3)
  ux = 150
  uy = 30
  If (r = 1) then
    ux = 30
    uy = 1000
  EndIf

  If (r = 2) Then
    ux = 1000
    uy = 1000
  EndIf

  x = (x + ux) / 2
  y = (y + uy) / 2

  GraphicsWindow.SetPixel(x, y, "LightGreen")
EndFor
```




Obrázek 35 – Trojúhelníkový fraktál

Když chcete opravdu vidět, jak tečky pomalu formují fraktál, můžete do cyklu přidat operaci **Program**.*Delay (počkat)*. Tato operace si vezme číslo, které určuje, kolik milisekund se má počkat. Zde je upravený program s tučně vyznačeným změněným řádkem.

```
GraphicsWindow.BackgroundColor = "Black"
x = 100
y = 100

For i = 1 To 100000
  r = Math.GetRandomNumber(3)
  ux = 150
  uy = 30
  If (r = 1) then
    ux = 30
    uy = 1000
  EndIf

  If (r = 2) Then
    ux = 1000
    uy = 1000
  EndIf

  x = (x + ux) / 2
  y = (y + uy) / 2

  GraphicsWindow.SetPixel(x, y, "LightGreen")
  Program.Delay(2)
EndFor
```

Zvýšení čísla program zpomalí. Experimentujte s čísly, abyste viděli, které vám bude nejvíce sedět.

Další změna, kterou můžete udělat, je zaměnit následující řádek:

```
GraphicsWindow.SetPixel(x, y, "LightGreen")
```

za

```
barva = GraphicsWindow.GetRandomColor()
```

```
GraphicsWindow.SetPixel(x, y, barva)
```

Tato změna způsobí, že program bude kreslit pixely náhodnými barvami.

*Kapitola 8*

# Želví grafika

## Logo

V 70. letech existoval velmi jednoduchý, ale užitečný programovací jazyk Logo, který byl používán několika vývojáři. Poté do něj kdosi přidal něco, čemu se říká „želví grafika“ a vymyslel „želvu“, která byla na obrazovce viditelná a odpovídala na příkazy jako *Vpřed, Doprava, Doleva,* atd. Pomocí želvy byli lidé schopni kreslit na obrazovku zajímavé tvary. Tímto se jazyk stal okamžitě přístupným a zajímavým pro lidi všech generací a způsobilo to jeho širokou popularitu v 80. letech.

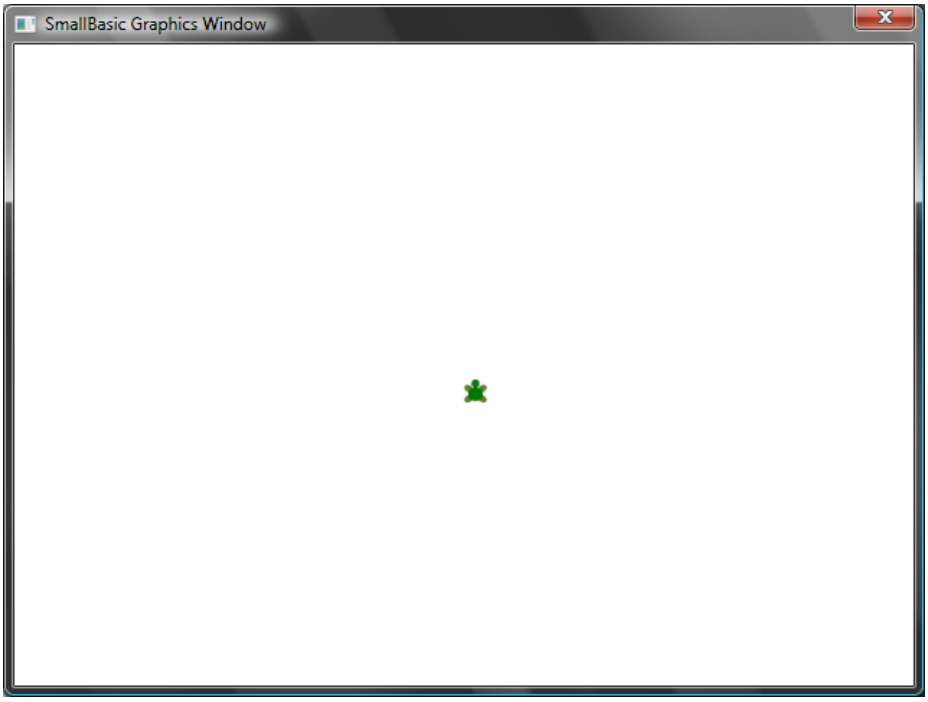
Small Basic přichází s objektem **Turtle (želva)** s mnoha příkazy. V této kapitole budeme používat želvu ke kreslení grafiky na obrazovku.

## Želva

Nejdříve musíme želvu zobrazit na obrazovce. K tomu stačí jediný řádek programu.

```
Turtle.Show()
```

Když spustíte tento program, zobrazí se bílé okno, stejné jako jsme viděli v jedné předchozí kapitole, až na to, že toto má ve svém středu želvu. Tato želva bude následovat naše instrukce a kreslit, co budeme chtít.




Obrázek 36 – Želva je viditelná

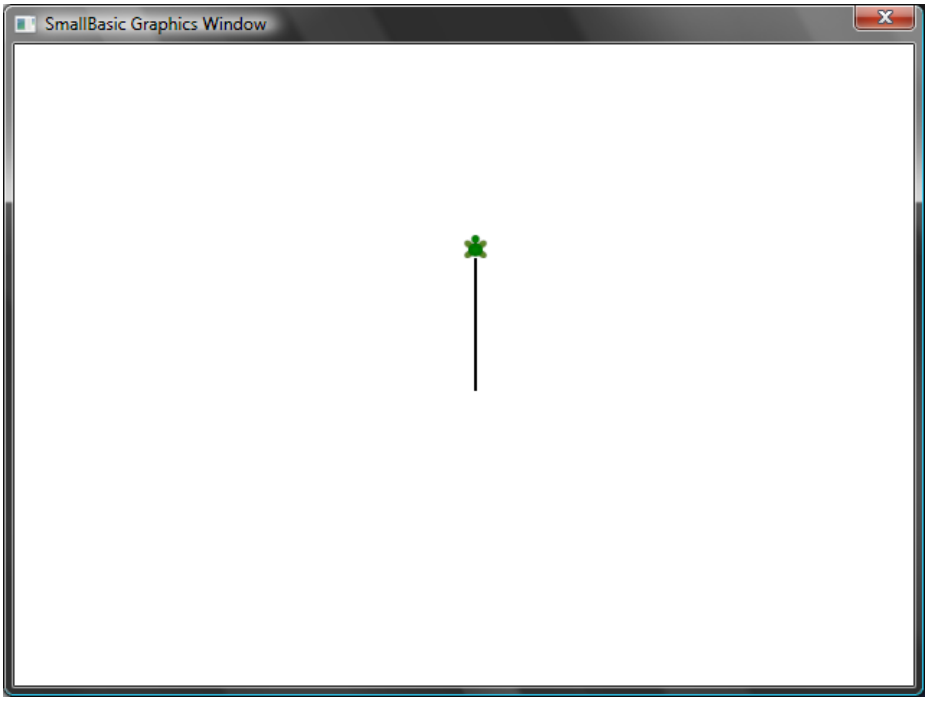
## Pohyb a kreslení

Jeden z příkazů, kterým želva rozumí, je **Move (pohyb)**. Tato operace vezme za vstup číslo. Toto číslo řekne želvě, jak daleko se má pohnout. V příkladě níže jí řekneme, aby se pohnula o 100 pixelů.

```
Turtle.Move(100)
```

Když spustíte tento program, můžete vidět, jak se želva pomalu pohybuje o 100 pixelů kupředu. Jak se pohybuje, můžete si všimnout čáry, kterou za sebou zanechává. Až se přestane pohybovat, uvidíte něco jako na obrázku níže.

*Pokud používáte operace s želvou, není nutné volat Show(). Želva se automaticky zobrazí, jakmile dostane nějakou instrukci.*



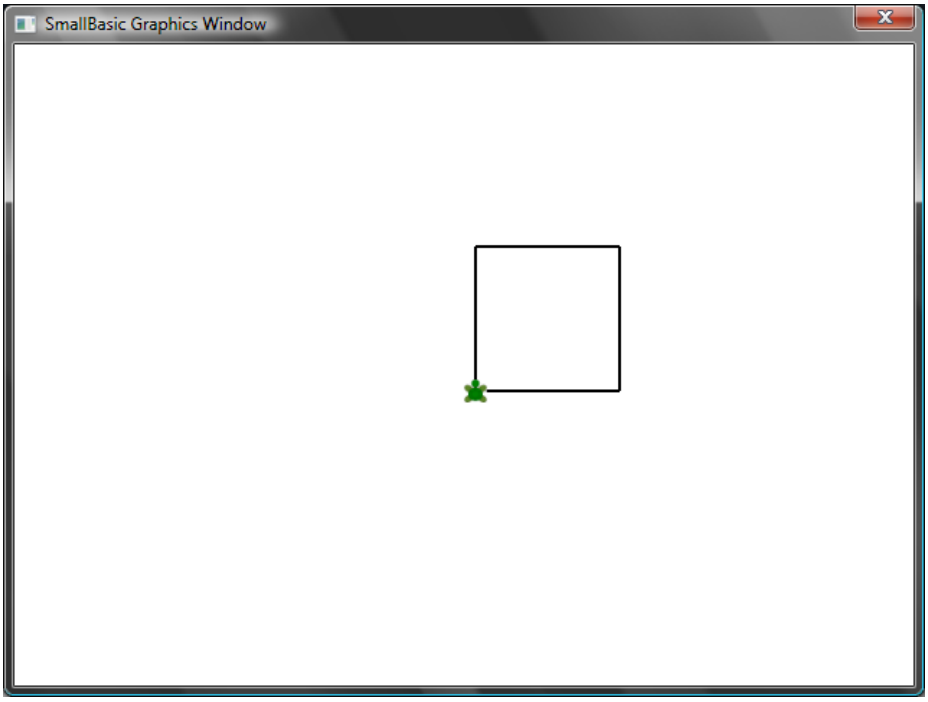

Obrázek 37 – Pohyb o sto pixelů

## Kreslení čtverce

Čtverec má dvě strany, dvě vodorovné a dvě svislé. Abychom jej nakreslili, musíme být schopni přimět želvu nakreslit čáru, otočit se doprava a to celé opakovat, dokud nebude čtverec hotov. Pokud bychom toto přeložili do programu, vypadal by takto:

```
Turtle.Move(100)
Turtle.TurnRight()
Turtle.Move(100)
Turtle.TurnRight()
Turtle.Move(100)
Turtle.TurnRight()
Turtle.Move(100)
Turtle.TurnRight()
```

Když spustíte tento program, uvidíte želvu kreslit čtverec a výsledek bude vypadat jako obrázek níže.




Obrázek 38 – Želva kreslící čtverec

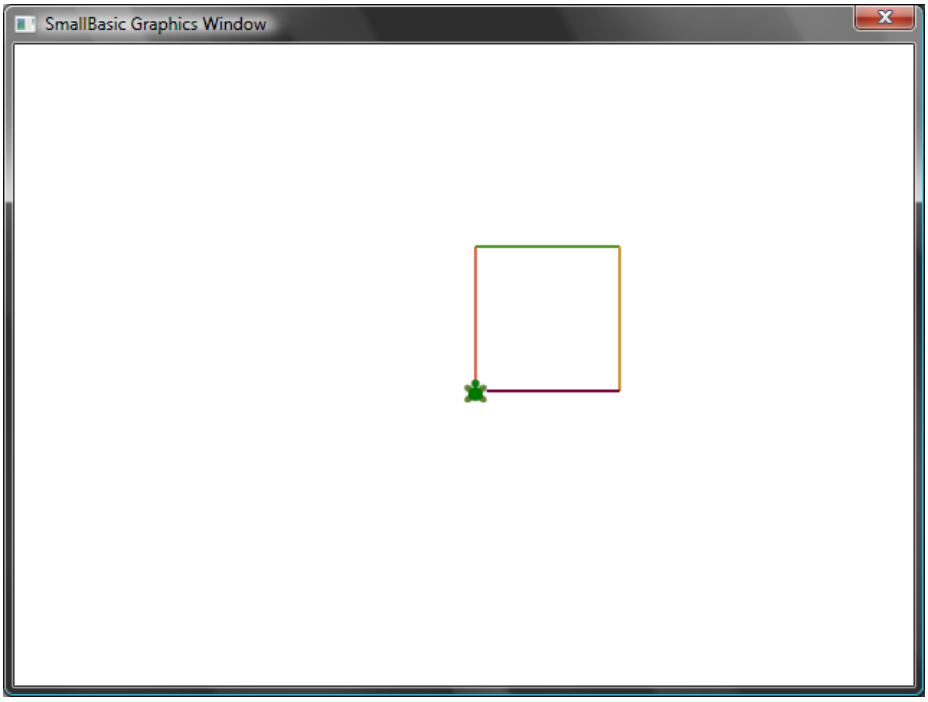
Bude zajímavé uvědomit si, že spouštíme pořád dokola dvě samé instrukce – čtyřikrát za sebou. A už jsme se naučili, že tyto opakující se příkazy mohou být spuštěny za pomoci cyklů. Takže, vezmeme program výše a upravíme ho, aby používal cyklus **For..EndFor**, a dostaneme mnohem jednodušší program.

```
 For i = 1 To 4
  Turtle.Move(100)
  Turtle.TurnRight()
EndFor
```

## Změna barev

Želva kreslí na úplně stejné grafické okno, které jsme viděli v předchozí kapitole. To znamená, že všechny operace, které jsme se v předchozí kapitole naučili, jsou zde stále platné. Například, následující program nakreslí čtverec s každou stranou jinak barevnou.

```
For i = 1 To 4
  GraphicsWindow.PenColor = GraphicsWindow.GetRandomColor()
  Turtle.Move(100)
  Turtle.TurnRight()
EndFor
```




Obrázek 39 – Změna barvy

## Kreslení komplexnějších tvarů

Želva má kromě **TurnRight (otočit doprava)** a **TurnLeft (otočit doleva)** ještě operaci **Turn (otočit)**. Tato operace má jeden vstup, který určuje úhel rotace. Použitím této operace je možné nakreslit jakýkoliv mnohoúhelník. Následující program nakreslí hexagon (pravidelný šestiúhelník).

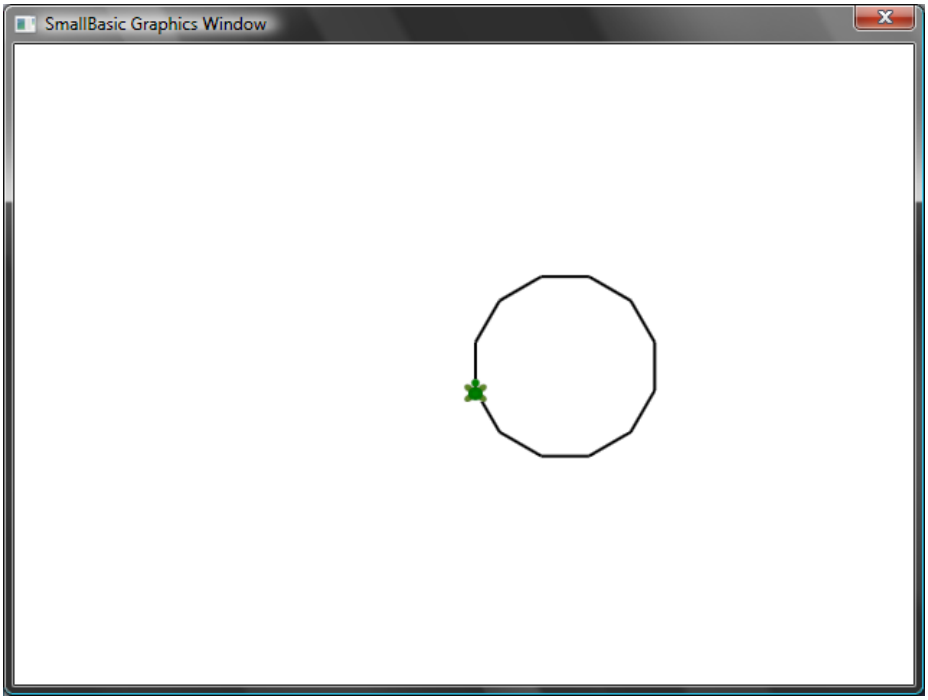
```
For i = 1 To 6
  Turtle.Move(100)
  Turtle.Turn(60)
EndFor
```

Vyzkoušejte tento program, abyste viděli, že opravdu vykreslí šestiúhelník. Všimněte si, že když úhel mezi stranami je 60 stupňů, použijete **Turn(60)**. Úhel mezi stranami pravidelného mnohoúhelníku můžeme získat vydělením 360 počtem stran. S touto informací a za použití proměnných můžeme snadno napsat celkem obecný program pro kreslení mnohoúhelníků.

```
strany = 12

delka = 400 / strany
uhel = 360 / strany
```

```
For i = 1 To strany
  Turtle.Move(delka)
  Turtle.Turn(uhel)
EndFor
```

Tímto programem můžete nakreslit jakýkoliv mnohoúhelník modifikací proměnné **strany**. Zadáním 4 dostaneme čtverec. Zadáním dostatečně velkého čísla, řekněme 50, dostaneme tvar nerozeznatelný od kružnice.




Obrázek 40 – Kreslení dvanáctiúhelníku

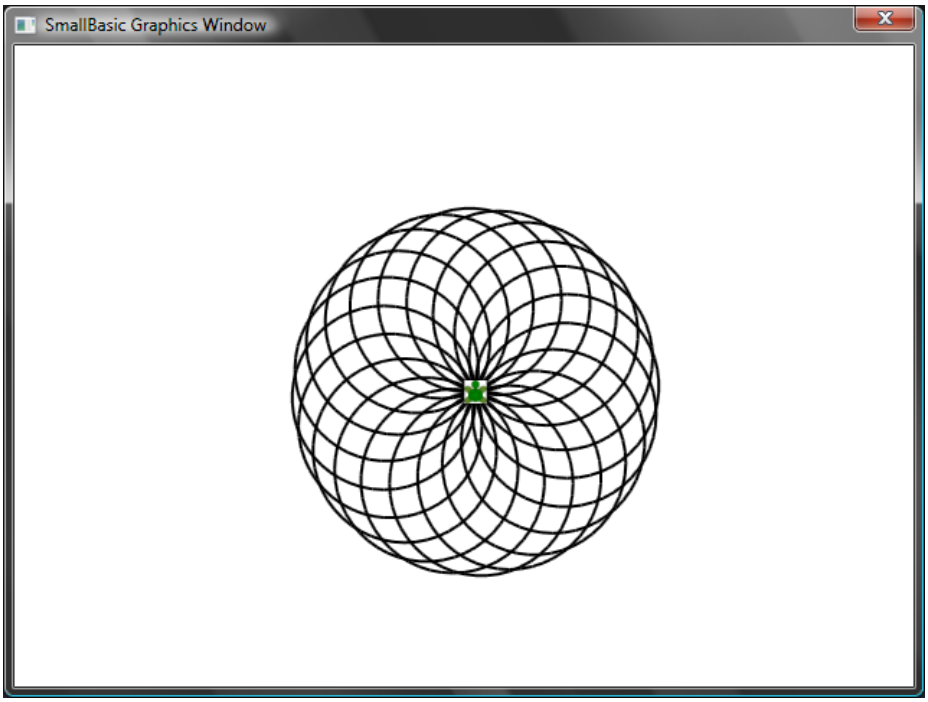
Za použití techniky, kterou jsme se právě naučili, můžeme přimět želvu kreslit několik kružnic, pokaždé s mírným otočením, což vyústí v zajímavý výstup.

```
strany = 50
delka = 400 / strany
uhel = 360 / strany

Turtle.Speed = 9

For j = 1 To 20
  For i = 1 To strany
    Turtle.Move(delka)
```

```
    Turtle.Turn(uhel)
  EndFor
  Turtle.Turn(18)
EndFor
```

Program výše má dva **For..EndFor** cykly, jeden v druhém. Vnitřní cyklus (*i=1 to strany*) je podobný polygonovému programu a je odpovědný za kreslení kružnice. Vnější cyklus (*j=1 to 20*) je odpovědný za otočení želvy o malý kousek po každé kružnici, která byla nakreslena. Toto řekne želvě, že má nakreslit 20 kružnic. Když to spojíme dohromady, dostaneme velmi zajímavý vzorek, jako ten níže zobrazený.

*V programu výše jsme želvu zrychlili nastavením Speed (rychlost) na 9. Tuto vlastnost můžete nastavit na hodnotu mezi 1 a 10, aby se želva pohybovala tak rychle, jak chcete.*



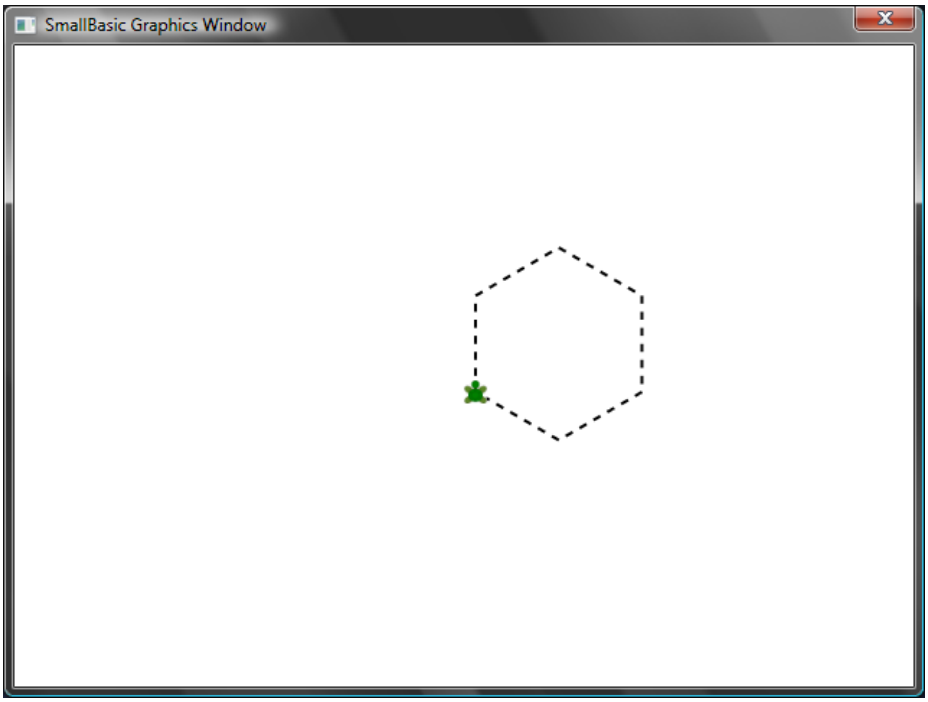

Obrázek 41 – Chození v kruzích

## Přerušování kreslení

Želvě můžete zakázat kreslit zavoláním operace **PenUp (zvednout pero)**. Toto vám umožní přesunout želvu kamkoli na obrazovku bez kreslení. Zavoláním **PenDown (položit pero)** začne želva opět kreslit. Toto může být použito k některým zajímavým efektům, jako například čárkované čáry. Zde je program, který nakreslí čárkovaný mnohoúhelník.

```
strany = 6

delka = 400 / strany
uhel = 360 / strany

For i = 1 To strany
  For j = 1 To 6
    Turtle.Move(delka / 12)
    Turtle.PenUp()
    Turtle.Move(delka / 12)
    Turtle.PenDown()
  EndFor
  Turtle.Turn(uhel)
EndFor
```

Tento program má opět dva cykly. Vnitřní nakreslí jednu čárkovanou čáru, zatímco ten vnější určuje, kolik čar se má nakreslit. V našem případě jsme použili 6 pro proměnnou **strany**, takže jsme dostali čárkovaný šestiúhelník, jak je ukázáno níže.




Obrázek 42 – Zvedání a pokládání pera

*Kapitola 9*

# Podprogramy

Při psaní programů se často dostaneme do situací, kdy budeme muset spouštět stále stejný shluk příkazů. V takových případech by pravděpodobně nemělo smysl opisovat několikrát stále stejné příkazy. Tehdy se hodí podprogramy.

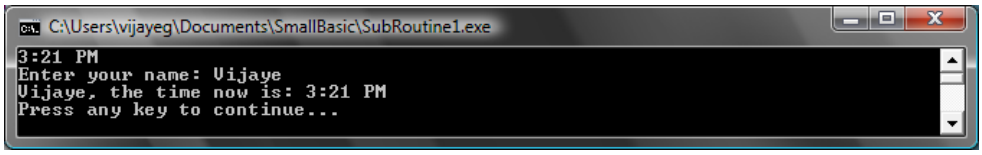
Podprogram je část kódu uvnitř většího programu, která obvykle dělá něco velmi specifického a může být zavolána odkudkoliv z programu. Podprogramy jsou označeny jménem, které se píše po klíčovém slovu **Sub** a jsou ukončeny slovem **EndSub**. Například, následující kód reprezentuje podprogram, jehož název je *VypisCas* a vypisuje na obrazovku čas.

```
Sub VypisCas
  TextWindow.WriteLine(Clock.Time)
EndSub
```

Dole je program, který obsahuje podprogram a volá jej z různých míst.

```
VypisCas()
TextWindow.Write("Zadejte své jméno: ")
jmeno = TextWindow.Read()
TextWindow.Write(jmeno + ", aktuální čas je: ")
VypisCas()

Sub VypisCas
  TextWindow.WriteLine(Clock.Time)
```

```
EndSub
```




Obrázek 43 – Volání jednoduchého podprogramu

Podprogram spustíte zavoláním *JmenoPodprogramu()*. Jako obvykle, závorky „()“ jsou nutné, aby počítač věděl, že chcete spustit podprogram.

## Výhody používání podprogramů

Jak jsme právě viděli, podprogramy pomáhají ke snížení množství kódu, který musíte napsat. Jak jednou napíšete podprogram *VypisCas*, můžete jej zavolat odkudkoliv z programu a vypsat aktuální čas.

*Pamatujte si, že můžete volat pouze podprogram z aktuálního programu. Nemůžete volat podprogram napsaný v jiném programu.*

Navíc, podprogramy vám pomáhají rozložit komplexní kód na menší části. Řekněme, že máte vyřešit složitou rovnici. Můžete napsat několik podprogramů, které řeší menší části celé rovnice. Poté můžete všechny výsledky spojit a dostat řešení celé rovnice.

Podprogramy také mohou pomoci v čitelnosti programu. Jinými slovy, pokud máte dobře pojmenované podprogramy pro běžně používané funkce programu, bude snazší váš program číst a porozumět mu. To je velmi důležité, pokud chcete rozumět cizímu programu nebo chcete, aby ostatní rozuměli vašemu programu. Někdy je to také dobré, pokud chcete přečíst svůj vlastní program, řekněme týden poté, co jste jej napsali.

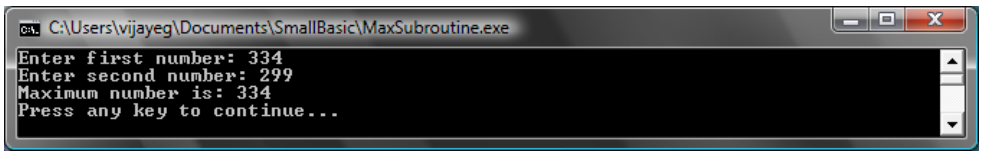
## Používání proměnných

Z podprogramu můžete přistoupit k jakékoliv proměnné z programu. Například, následující program načte dvě čísla a vypíše větší z nich. Všimněte si, že proměnná *max* se používá uvnitř i vně podprogramu.

```
TextWindow.Write("Zadejte první číslo: ")
cis1 = TextWindow.ReadNumber()
TextWindow.Write("Zadejte druhé číslo: ")
cis2 = TextWindow.ReadNumber()

NajdiMax()
```

```
TextWindow.WriteLine("Nejvyšší číslo je: " + max)

Sub NajdiMax
  If (cis1 > cis2) Then
    max = cis1
  Else
    max = cis2
  EndIf
EndSub
```

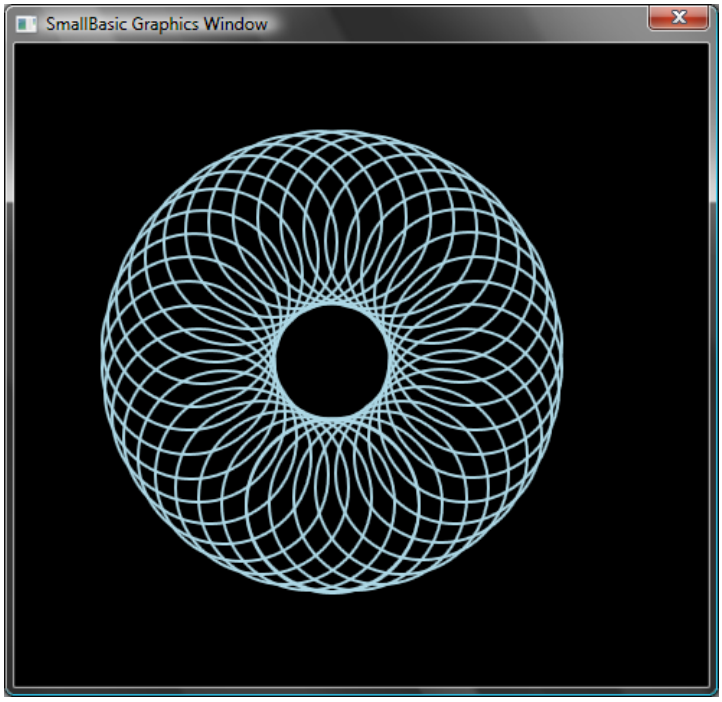
Výstup programu bude vypadat takto:




Obrázek 44 – Nejvyšší ze dvou čísel za pomoci podprogramu

Podívejme se na další příklad, který ilustruje použití podprogramů. Tentokrát použijeme grafický program, který spočítá mnoho bodů a bude je ukládat do proměnných *x* a *y*. Poté zavolá podprogram **NakreslitKruhStredem**, který nakreslí kruh při použití *x* a *y* jako střed.

```
GraphicsWindow.BackgroundColor = "Black"
GraphicsWindow.PenColor = "LightBlue"
GraphicsWindow.Width = 480
For i = 0 To 6.4 Step 0.17
  x = Math.Sin(i) * 100 + 200
  y = Math.Cos(i) * 100 + 200

  NakreslitKruhStredem()
EndFor

Sub NakreslitKruhStredem
  startX = x - 40
  startY = y - 40

  GraphicsWindow.DrawEllipse(startX, startY, 120, 120)
EndSub
```




Obrázek 45 – Grafický příklad pro podprogramy

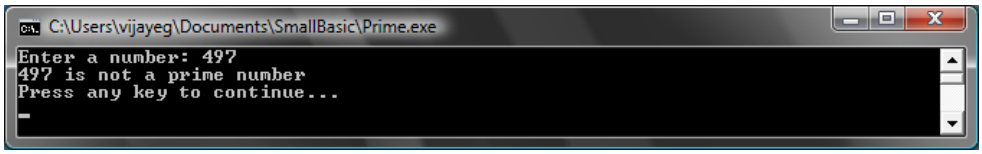
## Volání podprogramů uvnitř cyklů

Někdy jsou podprogramy volány z cyklů, přičemž pokaždé jsou spuštěny ty samé příkazy, ale s jinými hodnotami v jedné nebo více proměnných. Například řekněme, že máme podprogram nazvaný *ZkontrolujPrvocislo*, který zjistí, jestli dané číslo je prvočíslo. Můžete napsat program, který nechá uživatele zadat hodnotu a poté zjistí, zda je to prvočíslo nebo ne. V následujícím programu to můžete vidět.

```
TextWindow.Write("Zadejte číslo: ")
i = TextWindow.ReadNumber()
jePrvocislo = "Ano"
ZkontrolujProvcislo()
If (jePrvocislo = "Ano") Then
  TextWindow.WriteLine(i + " je prvočíslo")
Else
  TextWindow.WriteLine(i + " není prvočíslo")
EndIf

Sub ZkontrolujPrvocislo
  For j = 2 To Math.SquareRoot(i)
    If (Math.Remainder(i, j) = 0) Then
      jePrvocislo = "Ne"
```

```
      Goto Konec
    EndIf
  Endfor
Konec:
EndSub
```

Podprogram ZkontrolujPrvocislo vezme hodnotu *i* a pokouší se ji dělit menšími čísly. Pokud číslo vydělí *i* a nezanechá zbytek, potom *i* není prvočíslo. V této chvíli podprogram nastaví hodnotu *jePrvocislo* na „Ne“ a skončí. Pokud číslo nebylo dělitelné ani jedním menším číslem, hodnota *jePrvocislo* zůstává „Ano“.

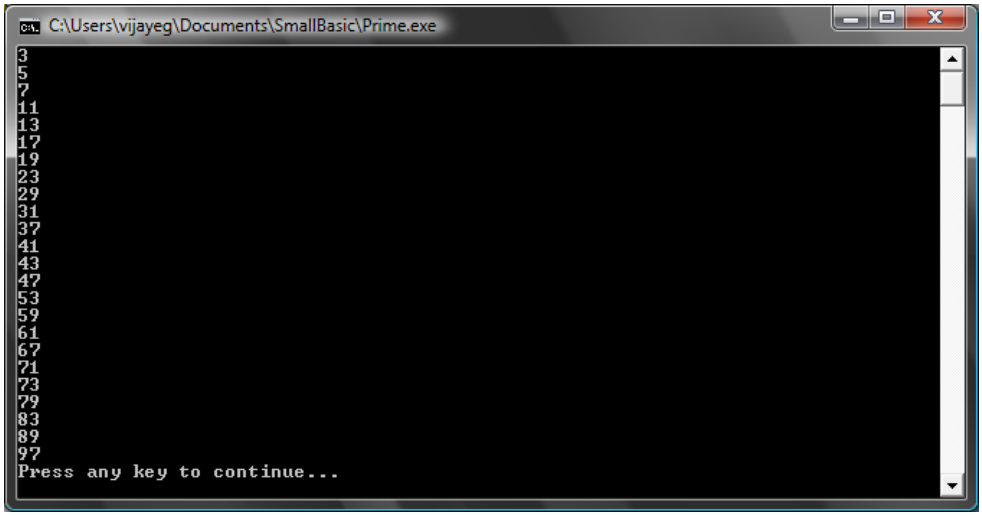



Obrázek 46 – Kontrola prvočísla

Teď, když máte podprogram, který otestuje, zda je číslo prvočíslo, můžete jej použít k vypsání všech prvočísel až do, řekněme, 100. Je opravdu snadné upravit program nahoře a přimět jej zavolat *ZkontrolujPrvocislo* uvnitř cyklu. Toto předá podprogramu jinou hodnotu pokaždé, když cyklus běží. Podívejme se, jak je to řešeno v příkladu níže.

```
For i = 3 To 100
  jePrvocislo = "Ano"
  ZkontrolujPrvocislo()
  If (jePrvocislo = "Ano") Then
    TextWindow.WriteLine(i)
  EndIf
EndFor

Sub ZkontrolujPrvocislo
  For j = 2 To Math.SquareRoot(i)
    If (Math.Remainder(i, j) = 0) Then
      jePrvocislo = "Ne"
      Goto Konec
    EndIf
  Endfor
Konec:
EndSub
```

V programu výše je hodnota *i* aktualizována pokaždé, když cyklus běží. Uvnitř cyklu je zavolán podprogram *ZkontrolujPrvocislo*. Tento podprogram pak vezme hodnotu *i* a spočítá, zda je to prvočíslo. Výsledek je uložen v proměnné *jePrvocislo*, ke které potom přistupuje cyklus mimo podprogram. Hodnota *i* je potom vypsána, pokud se ukáže, že je to prvočíslo. A protože cyklus začíná od trojky a končí ve stovce, dostaneme seznam všech prvočísel mezi 3 a 100. Níže je výsledek programu.




Obrázek 47 - Prvočísla

*Kapitola 10*

# Události a interaktivita

V prvních dvou kapitolách jsme si představili objekty, které mají *vlastnosti* a *operace*. Kromě vlastností a operací mají některé objekty něco, čemu se říká ***události***. Události jsou jako signály, které jsou spuštěny například jako odpověď na akce uživatele, jako pohyb nebo kliknutí myši. V jistém smyslu jsou události opakem operací. V případě operací je jako programátor zavoláte, pokud chcete, aby počítač něco udělal, zatímco v případě událostí vám počítač dá vědět, že se něco zajímavého stalo.

## Proč jsou události užitečné?

Události jsou důležité pro interaktivitu programu. Pokud chcete, aby uživatel mohl ovlivňovat běh vašeho programu, použijete právě události. Řekněme, že píšete nějakou hru. Chcete přece, aby si uživatel mohl vybrat, jak bude hrát, ne? Pak přichází na řadu události – získáte uživatelský vstup z programu využívajícího události. Zdá se to těžké, ale nebojte se, ukážeme si velmi jednoduchý příklad, který vám pomůže porozumět, co události vlastně jsou a jak mohou být použity.

Níže je velmi jednoduchý program, který má pouze jeden příkaz a jeden podprogram. Tento podprogram využije operaci *ShowMessage (zobraz zprávu)* na objektu GraphicsWindow, aby uživateli zobrazil zprávu.

```
GraphicsWindow.MouseDown = OnMouseDown

Sub OnMouseDown
  GraphicsWindow.ShowMessage("Kliknuli jste.", "Ahoj")
```

```
EndSub
```

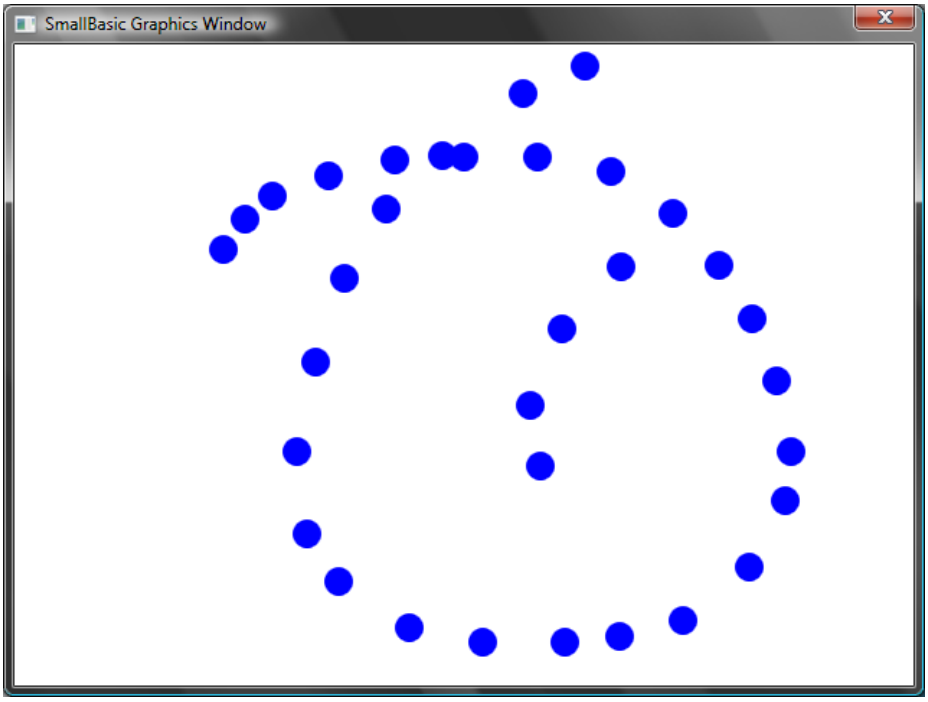
Zajímavá část programu je řádek, kde přiřazujete název podprogramu události **MouseDown (stisknutí tlačítka myši)** objektu GraphicsWindow. Všimnete si, že MouseDown vypadá velmi podobně jako vlastnost – až na to, že místo hodnoty jí přiřazujeme podprogram. Toto je na událostech velmi zvláštní – pokud událost nastane, podprogram je zavolán automaticky. V tomto případě je podprogram *OnMouseDown* zavolán pokaždé, když uživatel klikne na grafické okno. Spusťte program a vyzkoušejte jej. Pokaždé, když kliknete na grafické okno, uvidíte zprávu jako na obrázku dole.


Obrázek 48 – Odpověď na událost

Toto je velmi užitečné a umožňuje vám vytvářet velmi kreativní a zajímavé programy.

Můžete modifikovat podprogram *OnMouseDown*, aby dělal jiné věci, než zobrazil okno se zprávou. Například, jako v programu níže, můžete kreslit velké modré tečky všude, kam uživatel klikne.

```
GraphicsWindow.BrushColor = "Blue"
GraphicsWindow.MouseDown = OnMouseDown

Sub OnMouseDown
  x = GraphicsWindow.MouseX - 10
  y = GraphicsWindow.MouseY - 10
  GraphicsWindow.FillEllipse(x, y, 20, 20)
EndSub
```




Obrázek 49 - Handling Mouse Down Event

Všimněte si, že program výše používá *MouseX* a *MouseY*, aby zjistil souřadnice myši. Můžeme to potom použít k nakreslení kruhu za použití souřadnicí myši jako středu kruhu.

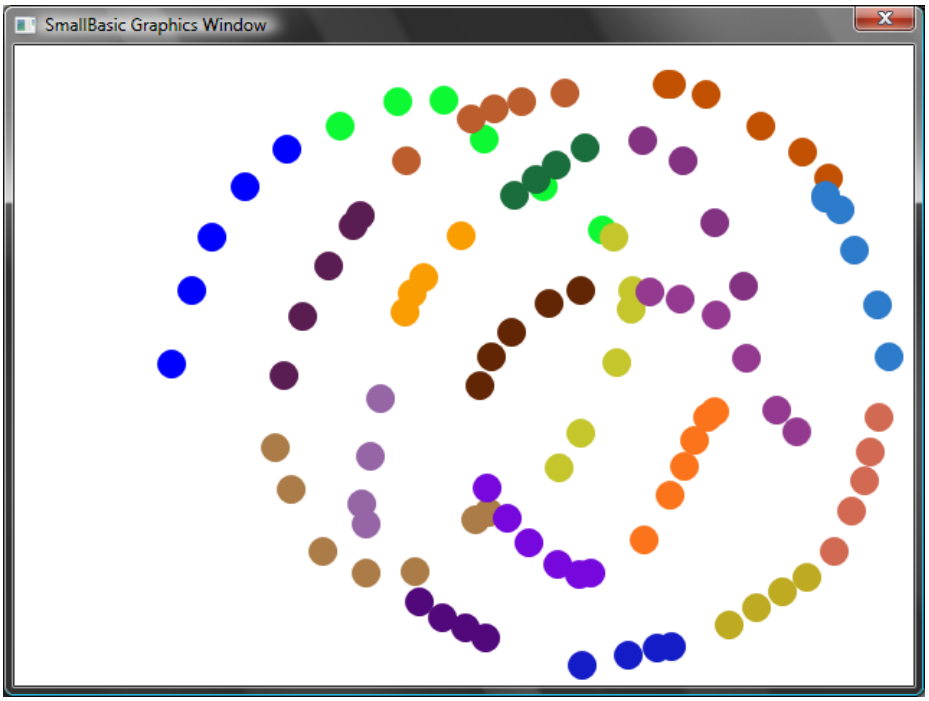
## Práce s více událostmi

Počet událostí, se kterými můžete pracovat, není omezen. Dokonce můžete mít jeden podprogram obsluhující více událostí. Nicméně, jedné události můžete přiřadit pouze jeden podprogram. Pokud se pokusíte přiřadit dva podprogramy stejné události, ta druhá vyhraje.

Abychom si toto ilustrovali, vezmeme si předchozí příklad a přidejte podprogram, který bude obsluhovat stisknutí kláves. Tento podprogram změní barvu štětce, takže když potom kliknete, dostanete jinak barevnou tečku.

```
GraphicsWindow.BrushColor = "Blue"
GraphicsWindow.MouseDown = OnMouseDown
GraphicsWindow.KeyDown = OnKeyDown

Sub OnKeyDown
  GraphicsWindow.BrushColor =
GraphicsWindow.GetRandomColor()
EndSub

Sub OnMouseDown
```

```
  x = GraphicsWindow.MouseX - 10
  y = GraphicsWindow.MouseY - 10
  GraphicsWindow.FillEllipse(x, y, 20, 20)
EndSub
```




Obrázek 50 – Práce s více událostmi

Pokud jste spustili tento program a klikli, dostali jste modrou tečku. Teď, pokud stisknete klávesu a znova kliknete, dostanete jinak barevnou tečku. Když stisknete klávesu, podprogram *OnKeyDown (stisknutí klávesy)* se spustí, což změní barvu štětce na náhodnou. Poté, co kliknete, se nakreslí kruh, ovšem s náhodně zvolenou barvou.

## Malování

Vyzbrojeni událostmi a podprogramy, můžeme teď napsat program, který uživateli umožní kreslit na okno. Je to překvapivě jednoduché, pokud si problém rozložíme na menší části. Jako první krok napišme program, který uživateli umožní pohybovat myší kdekoliv po grafickém okně a bude za sebou zanechávat stopu.

```
GraphicsWindow.MouseMove = OnMouseMove

Sub OnMouseMove
  x = GraphicsWindow.MouseX
  y = GraphicsWindow.MouseY
```

```
  GraphicsWindow.DrawLine(prevX, prevY, x, y)
  prevX = x
  prevY = y
EndSub
```

Nicméně pokud spustíte tento program, první čára vždy začíná z levého horního rohu okna (0,0). Tento problém můžeme vyřešit použitím události *MouseDown* a uložením hodnot *prevX* a *prevY* vždy, když událost nastane.

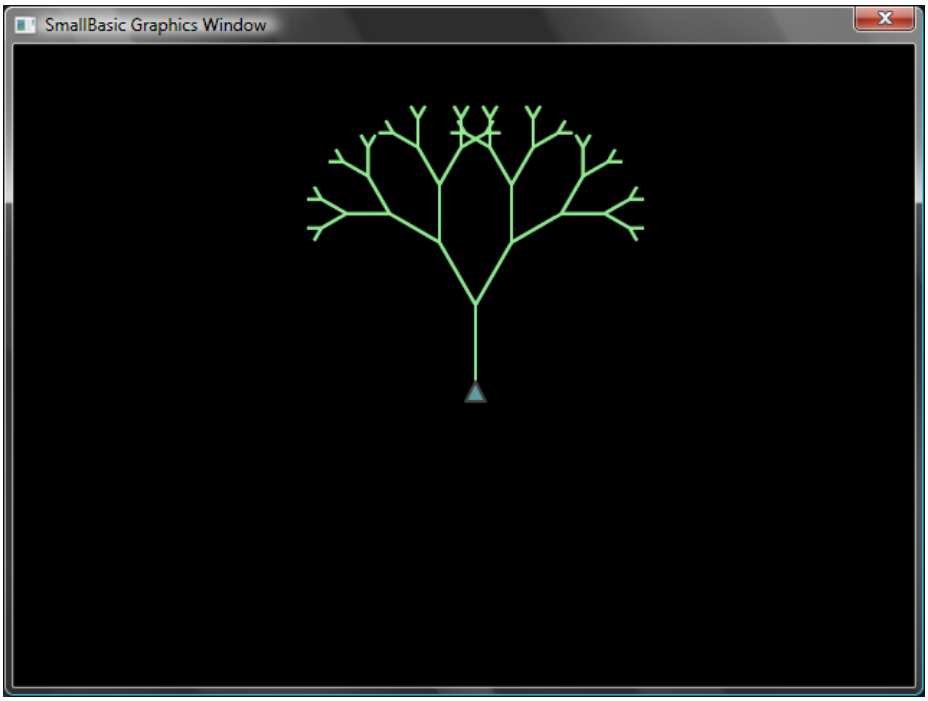
Také bychom chtěli, aby myš zanechávala stopu, pouze pokud uživatel drží tlačítko myši. Jinak by se čára kreslit neměla. Abychom tohoto chování docílili, použijeme vlastnost *IsLeftButtonDown (je stisknuto levé tlačítko)* objektu **Mouse (myš)**. Tato vlastnost zjistí, zda je levé tlačítko stisknuté nebo ne. Pokud ano, nakreslíme čáru, pokud ne, přeskočíme ji.

```
GraphicsWindow.MouseMove = OnMouseMove
GraphicsWindow.MouseDown = OnMouseDown

Sub OnMouseDown
  prevX = GraphicsWindow.MouseX
  prevY = GraphicsWindow.MouseY
EndSub

Sub OnMouseMove
  x = GraphicsWindow.MouseX
  y = GraphicsWindow.MouseY
  If (Mouse.IsLeftButtonDown) Then
    GraphicsWindow.DrawLine(prevX, prevY, x, y)
  EndIf
  prevX = x
  prevY = y
EndSub
```

*Příloha A*

# Zábavné příklady

## Želví fraktál




Obrázek 51 – Želva kreslící fraktál stromu

```
uhel = 30
delta = 10
vzdalenost = 60
Turtle.Speed = 9
```

```
GraphicsWindow.BackgroundColor = "Black"
GraphicsWindow.PenColor = "LightGreen"
KresliStrom()

Sub KresliStrom
  If (vzdalenost > 0) Then
    Turtle.Move(vzdalenost)
    Turtle.Turn(uhel)

    Stack.PushValue("vzdalenost", vzdalenost)
    vzdalenost = vzdalenost - delta
    KresliStrom()
    Turtle.Turn(-uhel * 2)
    KresliStrom()
    Turtle.Turn(uhel)
    vzdalenost = Stack.PopValue("vzdalenost")

    Turtle.Move(-vzdalenost)
  EndIf
EndSub
```

## Fotografie z Flickru

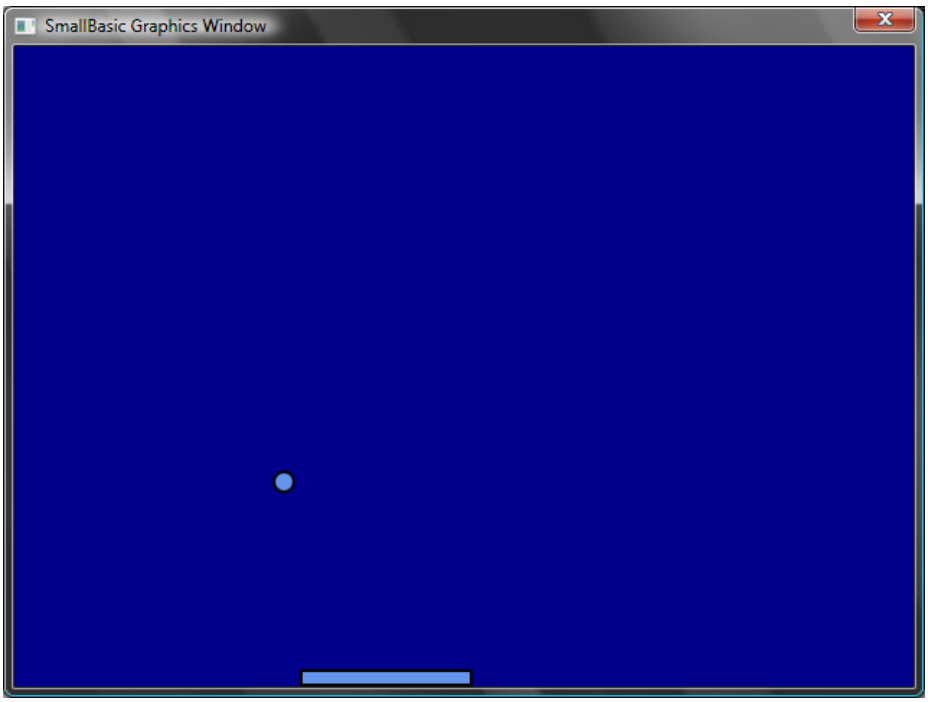

Obrázek 52 – Získávání obrázků z Flickru

```
GraphicsWindow.BackgroundColor = "Black"
GraphicsWindow.MouseDown = OnMouseDown

Sub OnMouseDown
    obr = Flickr.GetRandomPicture("mountains, river")
    GraphicsWindow.DrawResizedImage(obr, 0, 0, 640, 480)
EndSub
```

## Dynamická změna pozadí

```
For i = 1 To 10
  obr = Flickr.GetRandomPicture("mountains")
  Desktop.SetWallPaper(obr)
  Program.Delay(10000)
EndFor
```

## Arkanoid




Obrázek 53 - Arkanoid

```
GraphicsWindow.BackgroundColor = "DarkBlue"
plosina = Shapes.AddRectangle(120, 12)
```

```
micek = Shapes.AddEllipse(16, 16)
GraphicsWindow.MouseMove = OnMouseMove

x = 0
y = 0
deltaX = 1
deltaY = 1

Nahoru:
  x = x + deltaX
  y = y + deltaY

  gw = GraphicsWindow.Width
  gh = GraphicsWindow.Height
  If (x >= gw - 16 or x <= 0) Then
    deltaX = -deltaX
  EndIf
  If (y <= 0) Then
    deltaY = -deltaY
  EndIf

  padX = Shapes.GetLeft (plosina)
  If (y = gh - 28 and x >= padX and x <= padX + 120) Then
    deltaY = -deltaY
  EndIf

  Shapes.Move(micek, x, y)
  Program.Delay(5)

  If (y < gh) Then
    Goto Nahoru
  EndIf

GraphicsWindow.ShowMessage("Prohráli jste", "Arkanoid")

Sub OnMouseMove
  plosinaX = GraphicsWindow.MouseX
  Shapes.Move(plosina, plosinaX - 60, GraphicsWindow.Height
- 12)
EndSub
```

*Příloha B*

# Barvy

Zde je seznam pojmenovaných barev podporovaných jazykem Small Basic, kategorizovaných podle základního odstínu.

## Červené barvy

| | |
|---|---|
| IndianRed | #CD5C5C |
| LightCoral | #F08080 |
| Salmon | #FA8072 |
| DarkSalmon | #E9967A |
| LightSalmon | #FFA07A |
| Crimson | #DC143C |
| Red | #FF0000 |
| FireBrick | #B22222 |
| DarkRed | #8B0000 |

## Růžové barvy

| | |
|---|---|
| Pink | #FFC0CB |
| LightPink | #FFB6C1 |
| HotPink | #FF69B4 |
| DeepPink | #FF1493 |
| MediumVioletRed | #C71585 |
| PaleVioletRed | #DB7093 |

## Oranžové barvy

| | |
|---|---|
| LightSalmon | #FFA07A |
| Coral | #FF7F50 |
| Tomato | #FF6347 |
| OrangeRed | #FF4500 |
| DarkOrange | #FF8C00 |
| Orange | #FFA500 |

## Žluté barvy

| | |
|---|---|
| Gold | #FFD700 |
| Yellow | #FFFF00 |
| LightYellow | #FFFFE0 |
| LemonChiffon | #FFFACD |
| LightGoldenrodYellow | #FAFAD2 |
| PapayaWhip | #FFEFD5 |
| Moccasin | #FFE4B5 |

| | |
|---|---|
| PeachPuff | #FFDAB9 |
| PaleGoldenrod | #EEE8AA |
| Khaki | #F0E68C |
| DarkKhaki | #BDB76B |

## Fialové barvy

| | |
|---|---|
| Lavender | #E6E6FA |
| Thistle | #D8BFD8 |
| Plum | #DDA0DD |
| Violet | #EE82EE |
| Orchid | #DA70D6 |
| Fuchsia | #FF00FF |
| Magenta | #FF00FF |
| MediumOrchid | #BA55D3 |
| MediumPurple | #9370DB |
| BlueViolet | #8A2BE2 |
| DarkViolet | #9400D3 |
| DarkOrchid | #9932CC |
| DarkMagenta | #8B008B |
| Purple | #800080 |
| Indigo | #4B0082 |
| SlateBlue | #6A5ACD |
| DarkSlateBlue | #483D8B |
| MediumSlateBlue | #7B68EE |

## Zelené barvy

| | |
|---|---|
| GreenYellow | #ADFF2F |
| Chartreuse | #7FFF00 |
| LawnGreen | #7CFC00 |
| Lime | #00FF00 |
| LimeGreen | #32CD32 |
| PaleGreen | #98FB98 |
| LightGreen | #90EE90 |
| MediumSpringGreen | #00FA9A |
| SpringGreen | #00FF7F |
| MediumSeaGreen | #3CB371 |
| SeaGreen | #2E8B57 |
| ForestGreen | #228B22 |
| Green | #008000 |
| DarkGreen | #006400 |
| YellowGreen | #9ACD32 |
| OliveDrab | #6B8E23 |
| Olive | #808000 |
| DarkOliveGreen | #556B2F |
| MediumAquamarine | #66CDAA |
| DarkSeaGreen | #8FBC8F |
| LightSeaGreen | #20B2AA |
| DarkCyan | #008B8B |
| Teal | #008080 |

## Modré barvy

| | |
|---|---|
| Aqua | #00FFFF |
| Cyan | #00FFFF |
| LightCyan | #E0FFFF |
| PaleTurquoise | #AFEEEE |
| Aquamarine | #7FFFD4 |
| Turquoise | #40E0D0 |
| MediumTurquoise | #48D1CC |
| DarkTurquoise | #00CED1 |
| CadetBlue | #5F9EA0 |
| SteelBlue | #4682B4 |
| LightSteelBlue | #B0C4DE |
| PowderBlue | #B0E0E6 |
| LightBlue | #ADD8E6 |

| | |
|---|---|
| SkyBlue | #87CEEB |
| LightSkyBlue | #87CEFA |
| DeepSkyBlue | #00BFFF |
| DodgerBlue | #1E90FF |
| CornflowerBlue | #6495ED |
| MediumSlateBlue | #7B68EE |
| RoyalBlue | #4169E1 |
| Blue | #0000FF |
| MediumBlue | #0000CD |
| DarkBlue | #00008B |
| Navy | #000080 |
| MidnightBlue | #191970 |

## Hnědé barvy

| | |
|---|---|
| Cornsilk | #FFF8DC |
| BlanchedAlmond | #FFEBCD |
| Bisque | #FFE4C4 |
| NavajoWhite | #FFDEAD |
| Wheat | #F5DEB3 |
| BurlyWood | #DEB887 |
| Tan | #D2B48C |
| RosyBrown | #BC8F8F |
| SandyBrown | #F4A460 |
| Goldenrod | #DAA520 |
| DarkGoldenrod | #B8860B |
| Peru | #CD853F |
| Chocolate | #D2691E |
| SaddleBrown | #8B4513 |
| Sienna | #A0522D |
| Brown | #A52A2A |
| Maroon | #800000 |

## Bílé barvy

| | |
|---|---|
| White | #FFFFFF |
| Snow | #FFFAFA |
| Honeydew | #F0FFF0 |
| MintCream | #F5FFFA |
| Azure | #F0FFFF |
| AliceBlue | #F0F8FF |
| GhostWhite | #F8F8FF |
| WhiteSmoke | #F5F5F5 |
| Seashell | #FFF5EE |
| Beige | #F5F5DC |
| OldLace | #FDF5E6 |
| FloralWhite | #FFFAF0 |
| Ivory | #FFFFF0 |
| AntiqueWhite | #FAEBD7 |
| Linen | #FAF0E6 |
| LavenderBlush | #FFF0F5 |
| MistyRose | #FFE4E1 |

## Šedé barvy

| | |
|---|---|
| Gainsboro | #DCDCDC |
| LightGray | #D3D3D3 |
| Silver | #C0C0C0 |
| DarkGray | #A9A9A9 |
| Gray | #808080 |
| DimGray | #696969 |
| LightSlateGray | #778899 |
| SlateGray | #708090 |
| DarkSlateGray | #2F4F4F |
| Black | #000000 |