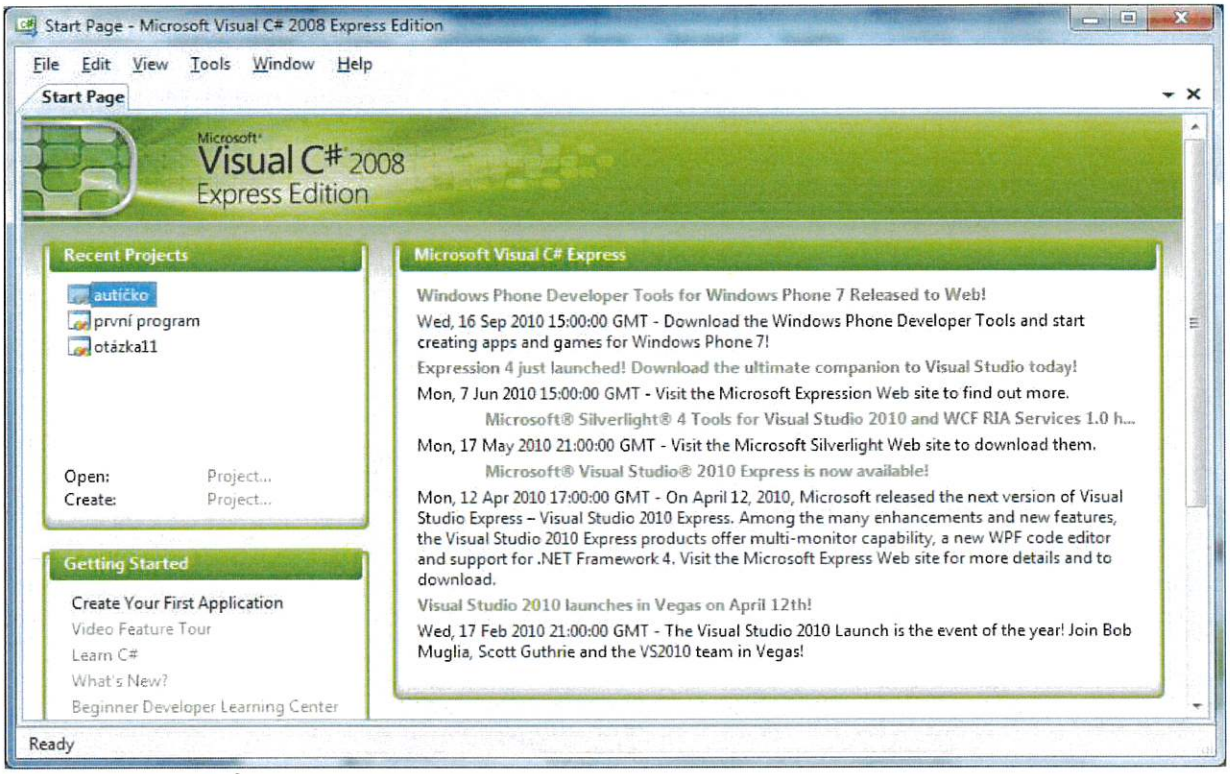
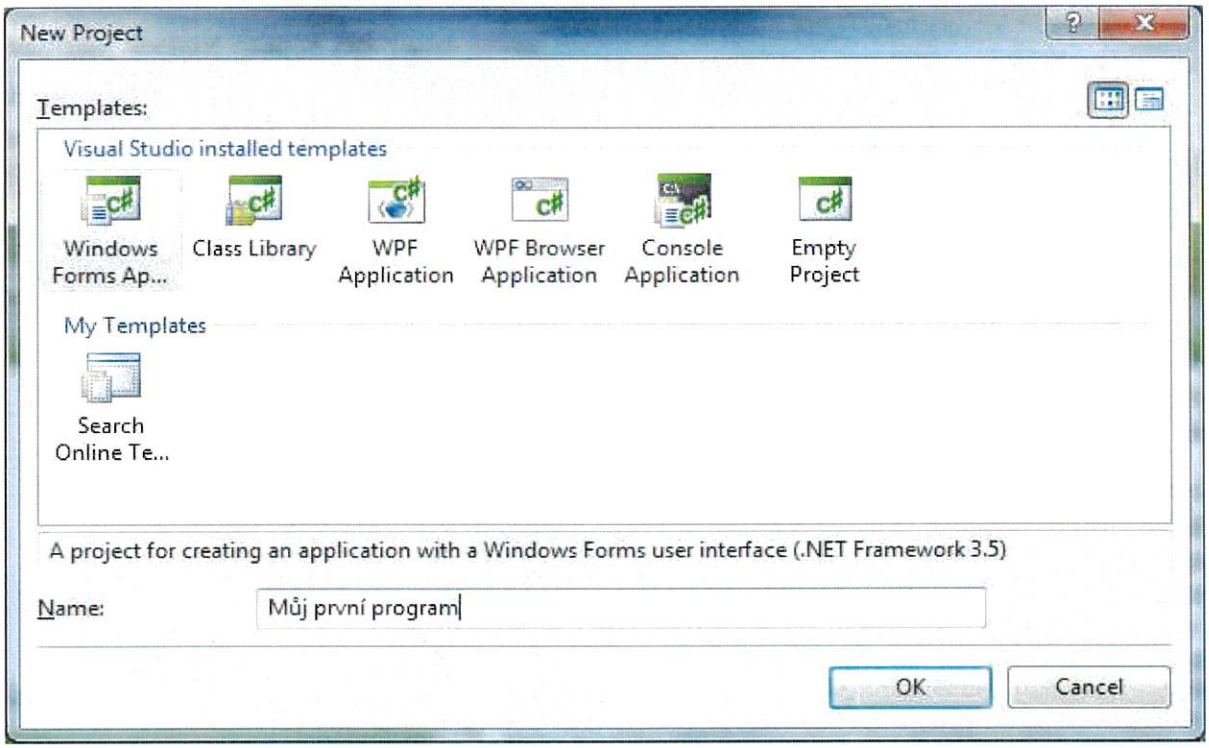
Programovací jazyk C# a vývojové prostředí

Po spuštění vývojového prostředí se objeví úvodní obrazovka:



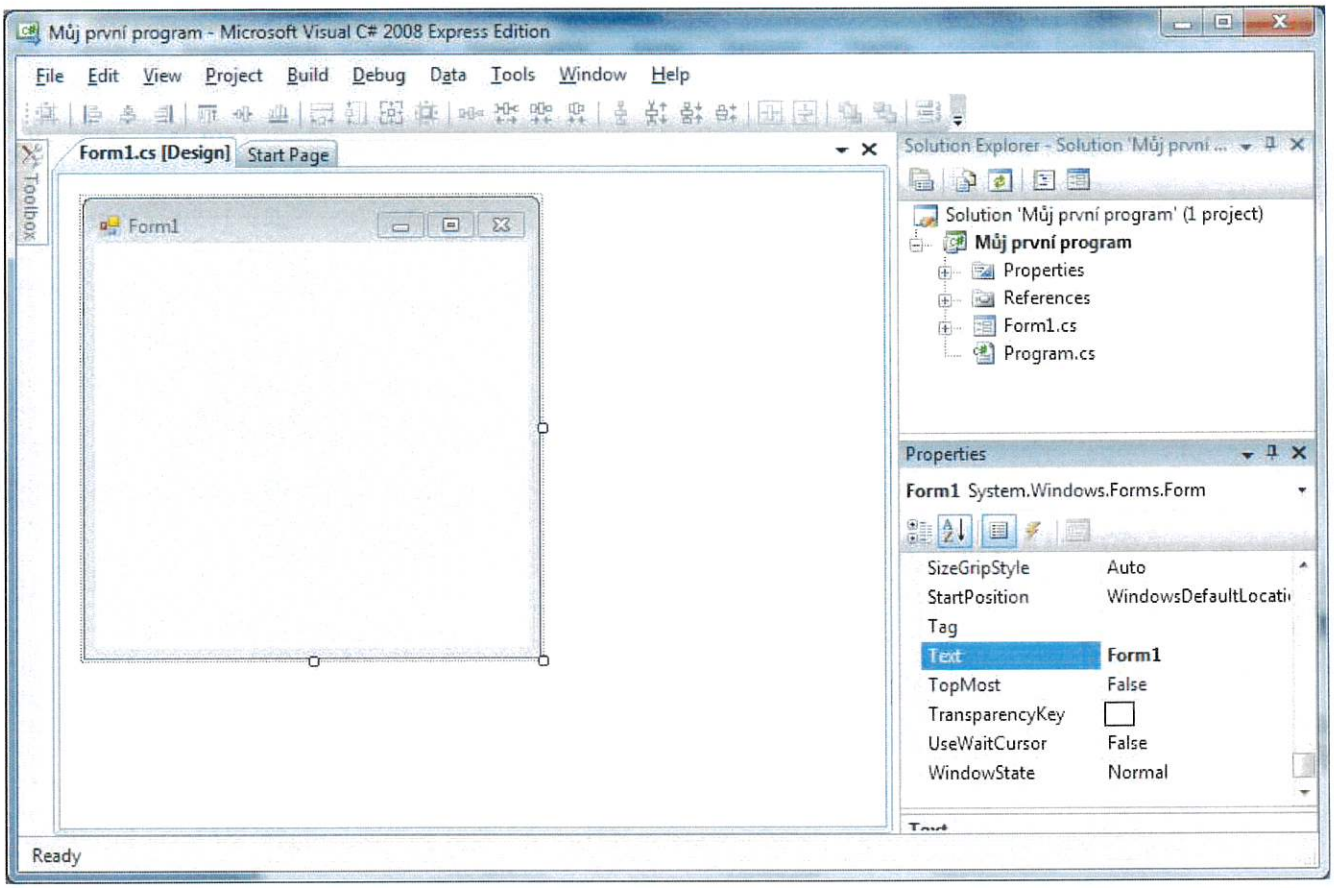
První program

Z nabídky *File* zvolte *New Project*. Otevře se dialogové okno:



Je třeba vybrat vhodnou šablonu podle toho, jaký druh programu budeme vytvářet. My budeme používat aplikace s grafickým uživatelským rozhraním, proto vybereme *Windows Forms Application*. V poli *Name:* zadáme jméno nového projektu a potvrdíme tlačítkem *OK*.

Zobrazí se nám tzv. ***povrch Designera***:



Povrch designera ukazuje navrhovaný vzhled programu. Úvodní obrazovka se nám schovala pod záložku *Start Page*.
Vpravo je **průzkumník řešení** (Solution Explorer), v kterém vidíme a můžeme otevírat jednotlivé součásti projektu (designera najdeme pod označením Form1.cs, Program.cs skrývá zdrojový kód programu). Pod ním je **editor vlastností a událostí** (Properties Windows), pokud vybereme konkrétní objekt, tak zde uvidíme vlastnosti případně události daného objektu. Vlevo je sada nástrojů (Toolbox), to je panel pro výběr ovládacích prvků. Pokud si některé z těchto oken zavřeme, znovu je otevřeme z nabídky *View*.
Okno, které vidíme v designeru je předlohou okna, v kterém poběží náš projekt. Název okna a jeho další vlastnosti lze měnit v editoru vlastností a událostí. Stačí vybrat okno programu a vpravo uvidíme výpis všech jeho vlastností. Na události přepneme tlačítkem s „bleskem“, ale to až později.
Vlastností je velké množství, stačí znát jen ty nejpoužívanější. U okna programu budeme vždy měnit název a text v titulní liště.
Název je vlastnost `(Name)` – automaticky tam bude Form1, my přepíšeme např na OknoProgramu.
U názvu objektu musí být jedno slovo bez diakritiky a bez mezer. Pro oddělení slov použijeme velká písmena.
Text v titulní liště je vlastnost `Text`. Tam napíšeme přesně to, co bude na titulní liště okna programu, samozřejmě česky a s mezerami.
Další často používané vlastnosti jsou:

- `BackColor` – barva pozadí okna
- `FormBorderStyle` – styl rámečku (např. jestli lze měnit velikost okna apod.)
- `MinimeBox, MaximeBox` – jestli má ikonku pro minimalizaci nebo maximalizaci
- `Size` – výchozí velikost okna

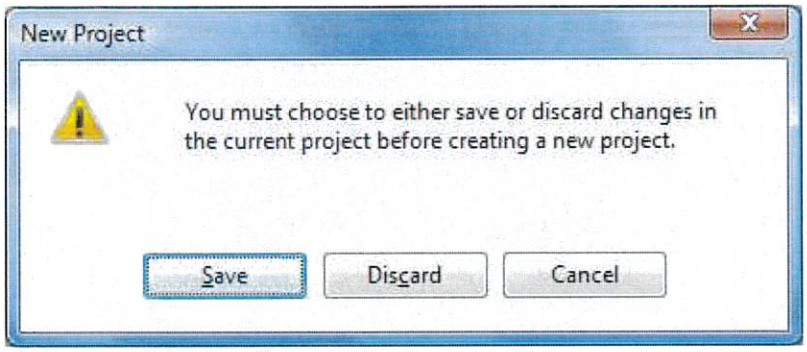
I když náš program ještě „nic nedělá“, můžeme ho spustit

Spuštění programu

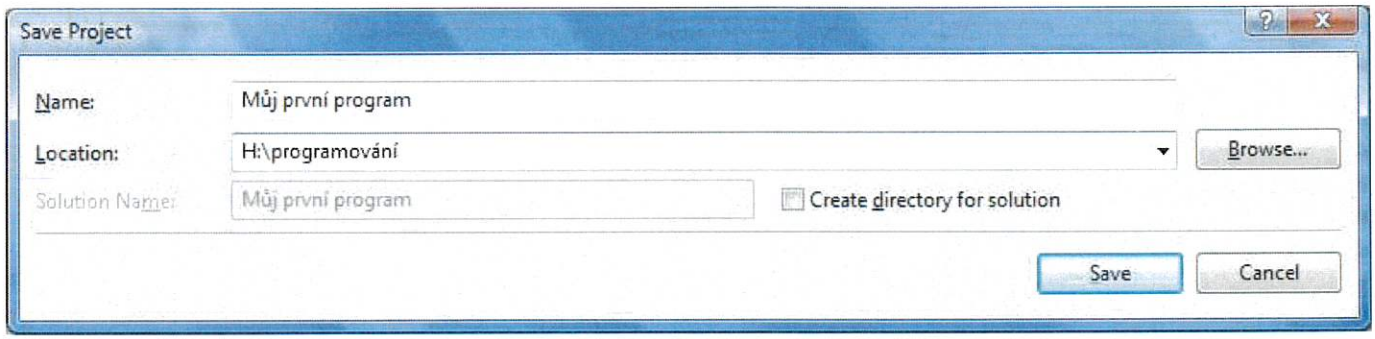
Program spustíme z nabídky *Debut - Start Debugging* nebo klávesou F5.
Po spuštění se objeví stejné okno, jako jsme viděli v designeru. Rozdíl mezi nimi je ten, že okno v designeru je „neživá předloha", jak bude program vypadat. Spuštěný program je skutečný, „živý". Je to samostatně existující okno, které lze zvětšovat, přemisťovat, atd. Ovládací prvky umístěné v tomto okně (až tam nějaké budou) fungují – tlačítka reagují na stisk, do textových polí lze psát, atd.
Než budeme pokračovat v sestavování programu v designeru, **musíme spuštěný program uzavřít křížkem.**

Uložení programu

Každý projekt bychom měli uložit. Buď vyčkáme, až budeme dotázáni, zda chceme projekt uložit – při zavírání vývojového prostředí nebo před spuštěním nového projektu, na dotaz odpovíme *Save* nebo uložíme sami z nabídky *File – Save All*. Při prvním ukládání se program ptá na umístění, pak už ukládá bez ptaní do stejného místa. Program je lepší ukládat průběžně. **K ukládání projektu nepoužívejte *Save As***, uložíte pouze jeden soubor, ale ne celý projekt!



Dotaz na uložení změn

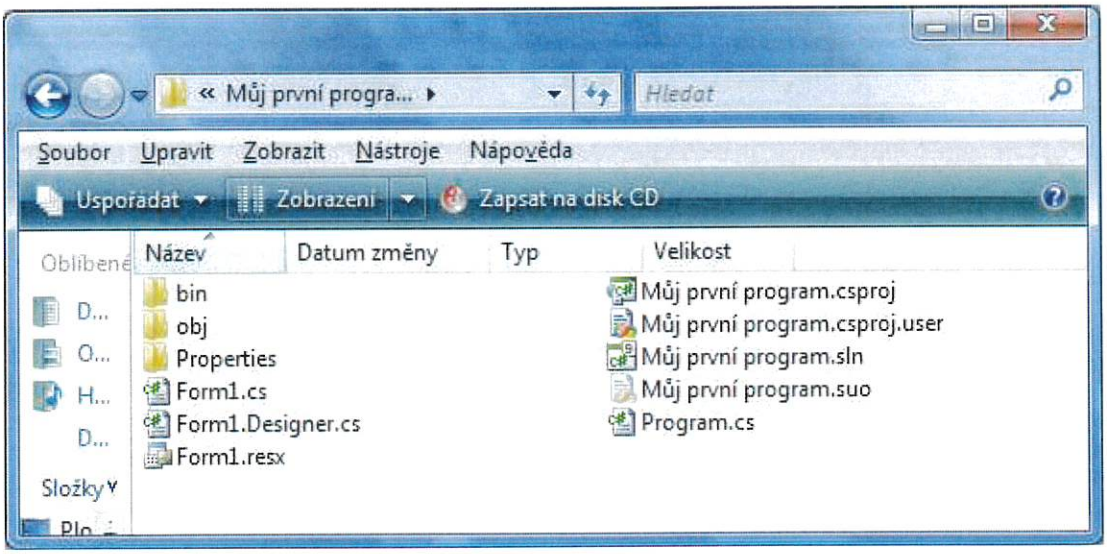


Dotaz na místo uložení

Při zadávání místa uložení zrušíme zaškrtnutí políčka *Create direktory for solution.*

Soubory projektu

Při ukládání projektu vznikne složka s názvem z řádku *Solution Name* a do ní se uloží všechny soubory projektu.
Soubory s příponou `.cs` jsou zdrojové texty programu, obsahují vlastní zápis programu v jazyce C#. Jsou ve formátu prostého textu a lze je editovat např. v poznámkovém bloku. Dále se ve složce nachází soubor s příponou `.sln` a `.csproj`. Pro začátečníka v programování není nutné vědět, k čemu slouží, stačí si pamatovat, že do nich nebudeme zasahovat.
Při kopírování projektu přeneseme vždy celou složku. Pro spuštění projektu poklepeme buď na soubor `.sln` nebo `.csproj` a projekt se otevře ve vývojovém prostředí.



Ovládací prvky

Ovládací prvky budeme do okna programu přetahovat z toolboxu (vlevo od okna programu). Prvky jsou uspořádány do skupin. Zpočátku budeme používat hlavně skupinu *Common Controls*.
Daný prvek zvolíme v toolboxu a do okna programu vložíme tažením levým tlačítkem myši. Stejný prvek můžeme dále vkládat kopírováním (Ctrl+c, Ctrl+v).
U všech prvků zadáme jejich název (Name). Pravidla pro zadávání názvů již známe od okna programu. Je vhodné prvky přejmenovávat a volit pojmenování, která napovídají, k čemu daný prvek slouží.
Při změně vlastností je třeba daný objekt vybrat myší, ale pozor, **vždy jen jedním kliknutím**. Pokud kliknete dvakrát, vyvoláte pro daný objekt obslužnou metodu hlavní události a otevře se vám okno, do kterého budete tuto metodu zapisovat. Toto okno je třeba zavřít a v editoru vlastností přepnout na události (tlačítko s bleskem). Danou událost najdeme, stiskneme na ní pravé tlačítko a vybereme *reset*. Pak je naše chyba napravená. **Nikdy sami nemažeme vložený kód prázdné metody**.
Následující ovládací prvky si vyzkoušíme na programu *Osobní údaje*

Textové pole - TextBox

Textové pole využíváme jak pro zadávání vstupních údajů, tak pro výpis výstupních údajů.
Vlastnosti textového pole:

`Text`	text v textovém poli
`BackColor`	barva pozadí textového pole
`ForeColor`	barva textu v textovém poli
`Font`	druh a velikost použitého písma
`MaxLength`	maximální počet znaků, které může uživatel do pole zadat
`Multiline`	při nastavení `True` umožňuje zadání víceřádkového textu
`ReadOnly`	při nastavení `True` není možné do pole zapisovat, ale pole je aktivní
`Enabled`	při nastavení `False` je pole neaktivní
`ScrollBars`	umožňuje okolo pole zobrazit posuvníky
`TextAlign`	zarovnání textu v poli
`UseSystemPasswordChar`	při nastavení `True` místo znaků zobrazuje puntíky
`WordWrap`	při nastavení `True` automaticky zalamuje text (musí být `Multiline`)

Popisek - Label

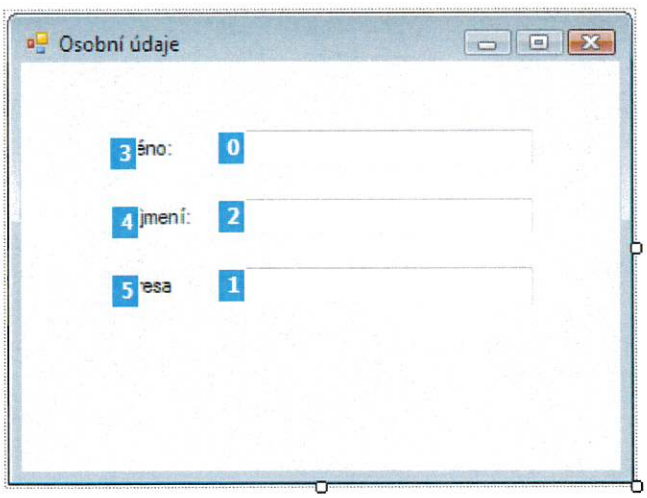
Popisek je nápis, který se nejčastěji umísťuje v blízkosti jiných ovládacích prvků a tím naznačuje jejich význam.
Vlastnosti popisku: `Text, backColor, ForeColor, Font`

Zarovnání ovládacích prvků

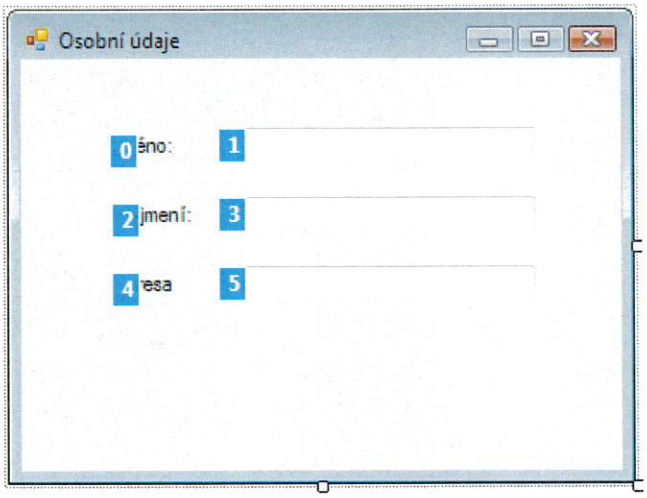
Pro zarovnání ve vodorovném směru využijeme modré vodící čáry, které nám vývojové prostředí samo nabízí. Pro vytvoření stejných mezer mezi prvky ve svislém směru musíme tyto prvky označit (pomocí klávesy Ctrl) a poté použít nabídku *Format – Vertical Spacing – Make Equal*
Pro vycentrování prvku na stránce použijeme nabídku *Format – Center in Form*.

Klávesové zkratky a pořadí tabulátoru

Popisku můžeme přiřadit klávesovou zkratku *Alt+písmeno*. Stačí do vlastnosti `text` napsat před dané písmeno znak &. Daný znak se v popisku podtrhne (někdy se podtržení objeví až po stisku klávesy *Alt*, záleží na nastavení Windows).
např. Objekt má jméno PopisekJméno a text &Jméno. Po spuštění programu pak stačí stisknout *alt+j* a kurzor skočí do textového pole za popiskem.
Aby toto fungovalo, musíme mít v pořádku pořadí tabulátoru, tj. pořadí v jakém se mi budou postupně aktivovat jednotlivé prvky při mačkání klávesy *Tab*. U každého prvku je vlastnost `TabIndex`, která určuje toto pořadí. Tento index se přiděluje podle pořadí, v jakém prvky do okna umísťujeme. Indexuje se od nuly. Abychom nemuseli od počátku myslet na toto pořadí nebo ho později nepřepisovali každému prvku zvlášť, můžeme využít nabídku *View – TabOrder* a zobrazit si tak pořadí tabulátoru. Klikáním na jednotlivá čísla hodnoty správně přečíslujeme. Čísla skryjeme tím, že znovu zvolíme *View – TabOrder*.



Původní pořadí tabulátoru



Správné pořadí tabulátoru

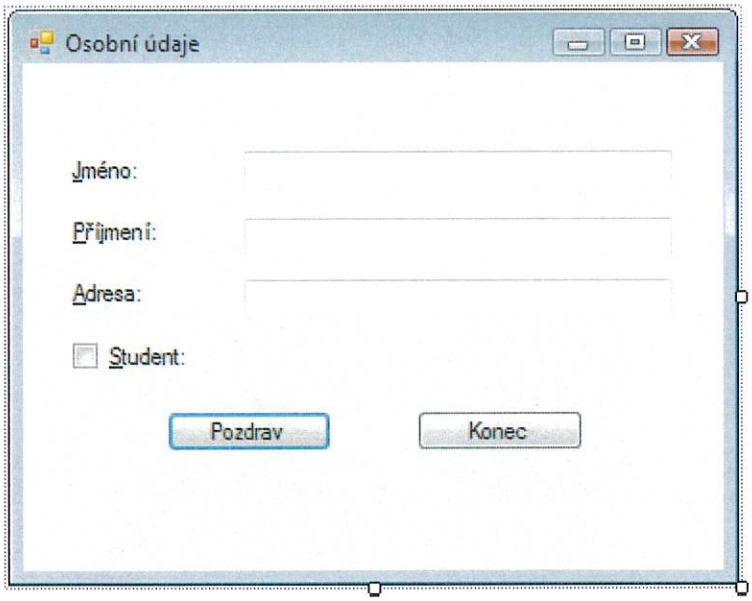
Zaškrtávací políčko - CheckBox

Zaškrtávací políčko je ovládací prvek pro dvoustavovou informaci ano/ne. Vlastnost `Text` určuje text popisku vedle zaškrtávacího políčka. Vlastnost `Checked` určuje zda je políčko zaškrtnuté (hodnota `True`) nebo není zaškrtnuté (hodnota `False`).

Tlačítko – Button

Tlačítko je zaoblený obdélník s textem, který po stisknutí spustí nějakou akci.
Vlastnost `Text` udává text napsaný na tlačítku.

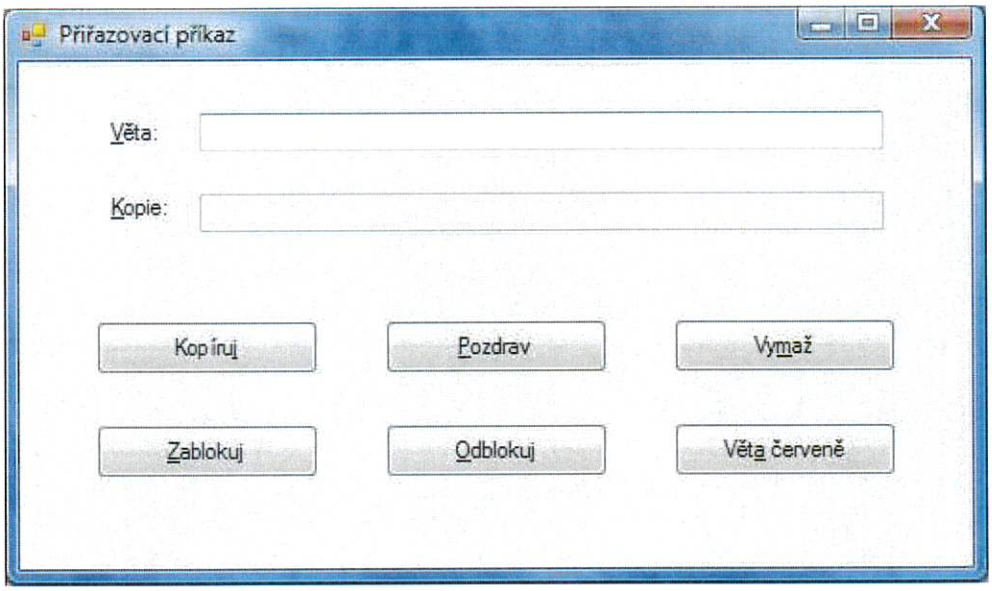
Náš program *Osobní údaje* vypadá jako na následujícím obrázku.



Pořadí tabulátoru je v pořádku, textová pole jsou aktivní, klávesové zkratky fungují, pouze tlačítko zatím po stisku nic nevyvolává. Abychom ho „oživili, musíme umět napsat obslužný program pro událost *Click*.“

Přiřazovací příkaz

Připravíme uživatelské rozhraní pro další program podle následujícího obrázku:



Nezapomeneme na klávesové zkratky Alt+písmeno a na pořadí tabulátoru.

Pro obsluhu události `click` použijeme u všech tlačítek ***přiřazovací příkaz***, který **hodnotu na pravé straně přiřadí na místo dané na levé straně.**

```
private void TlačítkoKopíruj_Click(object sender, EventArgs e)
{
    PoleKopie.Text = PoleVěta.Text;
}


private void TlačítkoPozdrav_Click(object sender, EventArgs e)
{
    PoleVěta.Text = "Ahoj lidi!";
}

private void TlačítkoVymaž_Click(object sender, EventArgs e)
{
    PoleVěta.Text = "";
}

private void TlačítkoČervená_Click(object sender, EventArgs e)
{
    PoleVěta.ForeColor = Color.Red;
}

private void TlačítkoZablokuj_Click(object sender, EventArgs e)
{
    PoleVěta.Enabled = false;
}

private void TlačítkoOdblokuj_Click(object sender, EventArgs e)
{
    PoleVěta.Enabled = true;
}
```

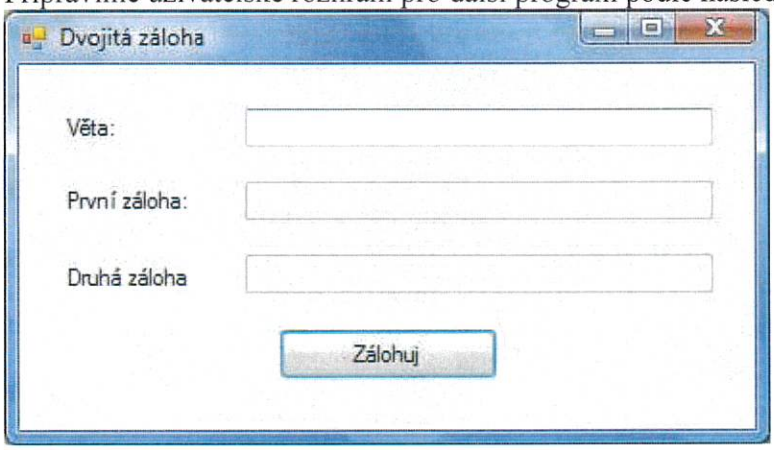
Více příkazů v metodě

Dosud jsme měli v každé metodě pouze jeden příkaz. Nemusí to tak být vždy.
Vylepšíme si tlačítka *Zablokuj* a *Odblokuj*, že tlačítko, které v danou chvíli nemá smysl použít, bude zablokované.

```
private void TlačítkoZablokuj_Click(object sender, EventArgs e)
{
    PoleVěta.Enabled = false;
    TlačítkoZablokuj.Enabled = false;
    TlačítkoOdblokuj.Enabled = true;
}

private void TlačítkoOdblokuj_Click(object sender, EventArgs e)
{
    PoleVěta.Enabled = true;
    TlačítkoZablokuj.Enabled = true;
    TlačítkoOdblokuj.Enabled = false;
}
```

Sekvenční zpracování

Připravíme uživatelské rozhraní pro další program podle následujícího obrázku:



Pokud napíšeme obslužnou metodu události `click` tlačítka *Kopíruj* následujícím způsobem, zjistíme, že nejde o dvojitou zálohu, ale do obou polí se kopíruje totéž.

```
private void TlačítkoZálohuj_Click(object sender, EventArgs e)
{
    PolePrvníZáloha.Text = PoleVěta.Text;
    PoleDruháZáloha.Text = PolePrvníZáloha.Text;
}
```

Aby se do druhé zálohy kopíroval obsah první, musíme příkazy přehodit.

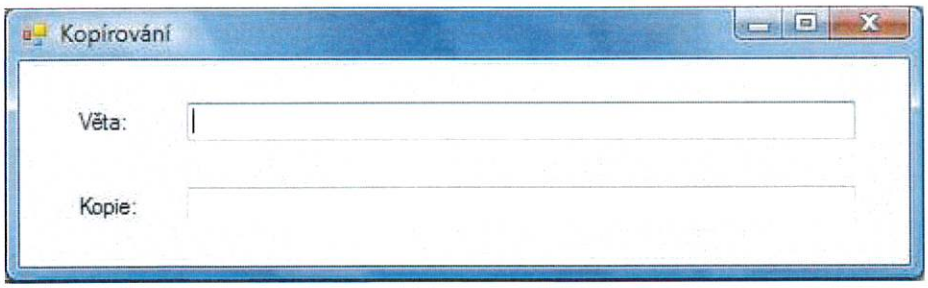
```
private void TlačítkoZálohuj_Click(object sender, EventArgs e)
{
    PoleDruháZáloha.Text = PolePrvníZáloha.Text;
    PolePrvníZáloha.Text = PoleVěta.Text;
}
```

Příkazy v metodě se provádí sekvenčně, tj. jeden za druhým v pořadí, jak jsou napsány.

Příklady na procvičení:

Příklad 1: Do programu *přiřazovací příkaz* přidejte tlačítko *Obnov původní stav*, které nastaví zpět výchozí barvu textu, vymaže obě textová pole, zablokuje tlačítko *Odblokuj* a odblokuje tlačítko *Zablokuj*.

Příklad 2: Napište program *Kopírování* podle následujícího obrázku. Kopie věty se bude vytvářet hned při psaní, tj. při jakékoliv změně v poli *Věta*.



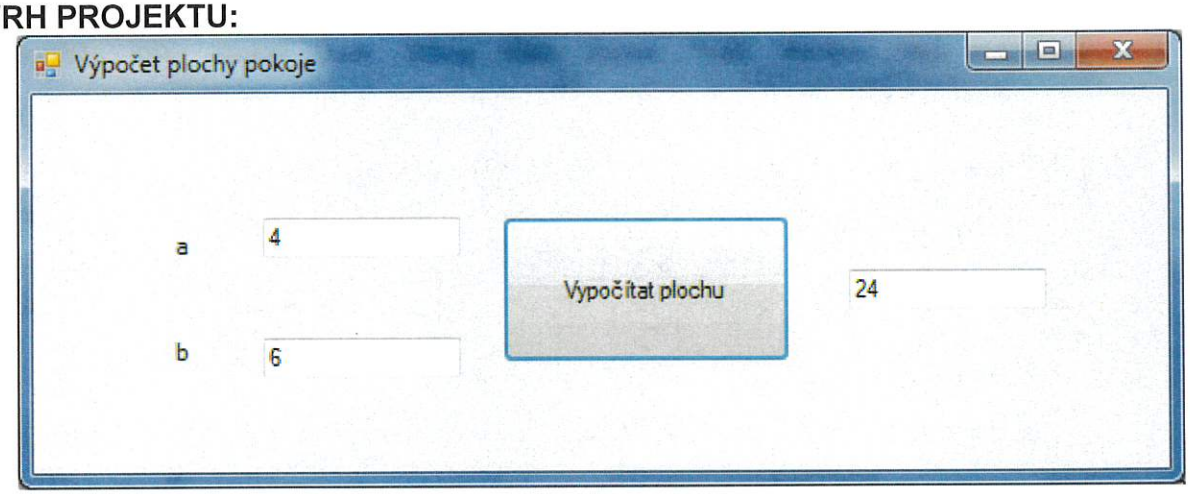
Příklad 3: Napište program *Schovávaná* s jedním tlačítkem *Schovej se* , které schová celý program na dvě vteřiny. Použijte příkazy `Hide();` a `Show();` a na „uspání" programu na dvě vteřiny použijte `System.Threading.Thread.Sleep(2000);`
Program dále vylepšete tak, aby tlačítko bylo žluté a zčervenalo, když nad ním bude myš. Použijte události *MouseEnter* a *MouseLeave*
Dále přidejte tlačítko *Pípej*, které vyvolá pípnutí, viz příkaz `System.Media.SystemSounds.Beep.Play();`

Práce s číselnými proměnnými

ŘEŠENÝ PŘÍKLAD

ZADÁNÍ:
Zadejte rozměry pokoje: strany a, b. Vypočítejte, kolik musíte koupit m^2 plovoucí podlahy na pokrytí celé místnosti.

NÁVRH PROJEKTU:

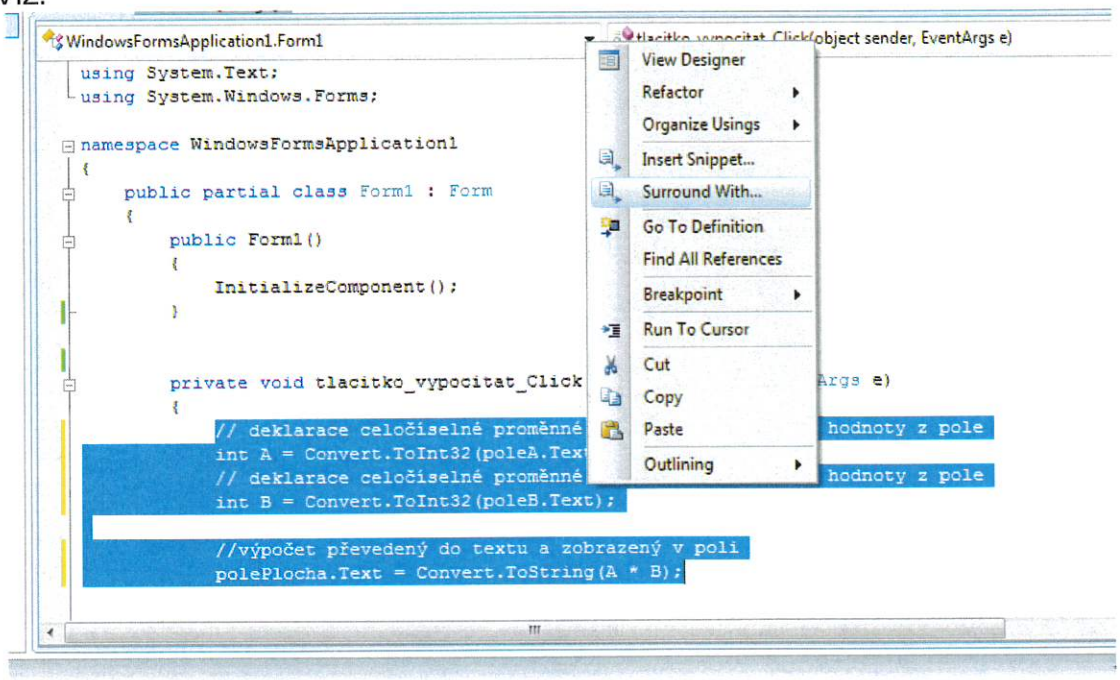


KÓD ŘEŠENÍ:

```
namespace práceSCisly
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }


        private void tlacitko_vypocitat_Click(object sender, EventArgs e)
        {
            // deklarace celočíselné proměnné A a převedení zadané hodnoty z pole
            int A = Convert.ToInt32(poleA.Text);
            // deklarace celočíselné proměnné B a převedení zadané hodnoty z pole
            int B = Convert.ToInt32(poleB.Text);

            //výpočet převedený do textu a zobrazený v poli
            polePlocha.Text = Convert.ToString(A * B);
        }
    }
}
```

ZÁKLADNÍ PROBLÉM PŘI PRÁCI S ČÍSLY:
TextBox umožňuje přes vlastnost Text zadat vstupní hodnotu **pouze ve formě textu**, takže když potřebujeme tuto hodnotu nastavit celočíselné proměnné A (typ int), musíme jí nejprve převést z textu na číselný typ **int**. K tomu použijeme metodu **Convert**, po stisknutí **.** dostaneme nabídku, do kterého typu chceme převést (např. ToInt32. ToString.), použijeme **ToInt32.**
Podobně musíme postupovat i po výpočtu, kdy máme číselný výsledek, který potřebujeme zobrazit jako text. Opět použijeme metodu **Convert,** tentokrát ale převádíme z čísla do textu, tedy **ToString**

Další problém může nastat, když nebude zadaná číselná hodnota ve vstupních polích před stiskem tlačítka Vypočítat plochu. Běh programu se zastaví hned u prvního převodu a systém oznámí chybu. Nejprve rada – běžící aplikaci zastavíme stiskem **Stop debugging** v nabídce hlavního menu **Debug** (není nutné použít CTRL+ALT+DEL).
Řešení daného problému je použití příkazu **vyjímky – try.** Příkaz nemusíme psát, ale využijeme připravených nástrojů. Vybereme část kódu, kterou má počítač zkusit provést a tu obkroužíme příkazem try. Do části **catch** vložíme příkazy, které se mají vykonat v případě chyby. Viz:




```
private void tlacitko_vypocitat_Click(object sender, EventArgs e)
    {
        try
         {
          // deklarace celočíselné proměnné A a převedení zadané hodnoty z pole
         int A = Convert.ToInt32(poleA.Text);
          // deklarace celočíselné proměnné B a převedení zadané hodnoty z pole
         int B = Convert.ToInt32(poleB.Text);

          //výpočet převedený do textu a zobrazený v poli
         polePlocha.Text = Convert.ToString(A * B);
         }
        catch
            {
                MessageBox.Show("Zadejte dvě celočíselné hodnoty");
            }
        }
```

TYPOVÉ PŘÍKLADY:

1. Vypočítejte obvod trojúhelníka ze zadaných (pomocí textBoxu) tří stran, vypište výsledek.

2. Zadejte cenu jedné vstupenky a počet, který chcete koupit. Po stisknutí tlačítka Zaplatit vypište celkovou cenu.

3. Zadejte, kolik peněz máte v peněžence a kolik stál nákup. Po stisku tlačítka vypište, kolik vám zůstane.

Grafika

Napíšeme program, ve kterém vykreslíme do okna programu několik úseček a obrazců (nevyplněných).

Událost Paint

Abychom mohli kreslit do okna programu, musíme pro něj napsat obslužnou metodu události `Paint`.
Událost paint je pro okno programu vyvolána nejčastěji z následujících příčin:

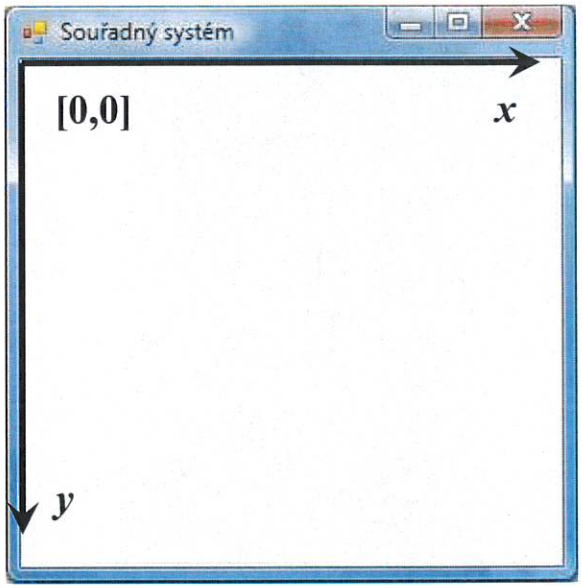
- Něž se okno porvé vykreslí (po spuštění programu)
- Když se okno minimalizuje na hlavní panel a poté obnoveno
- Jestliže je část okna něčím překryta a uživatel ji odkryje

Kreslící plocha

Obrazce musíme kreslit „někam", tedy na nějakou kreslící plochu. Jako název pro kreslící plochu budeme používat název `kp`. Tuto proměnnou deklarujeme a zároveň inicializujeme příkazem

```
Graphicskp = e.Graphics;
```

Souřadný systém okna



Levý horní roh okna má souřadnice [0,0].*x*-ová souřadnice roste směrem doprava a *y*-ová roste směrem dolů (opačně než známe z matematiky). Hodnoty se udávají v pixelech, není-li uvedem jiná jednotka

Pera

Prvním parametrem metod pro kreslení obrazce je vždy *pero*. Pero je úplnou charakteristikou čáry, tj. kombinací barvy, tloušťky a stylu.
Pro začátek nám bude stačit tenká plná čára, kterou si můžeme vybrat ze „sady per" `Pens`. Napíšeme `Pens.`*barva_čáry*. Např `Pens.Red`.

Grafické příkazy

Obrazce se kreslí na kp, proto píšeme `kp.`*metoda_pro_kreslení_obrazce*(*parametry*);

Metoda	*Funkce*	*Parametry*
`DrawLine`	kreslí úsečku	pero, souřadnice koncových bodů
`DrawRectangle`	kreslí obdélník	pero, souřadnice levého horního rohu, šířka, výška
`DrawEllipse`	kreslí elipsu	pero, souř. levého hor. rohu opsaného obdélníka, šířka, výška

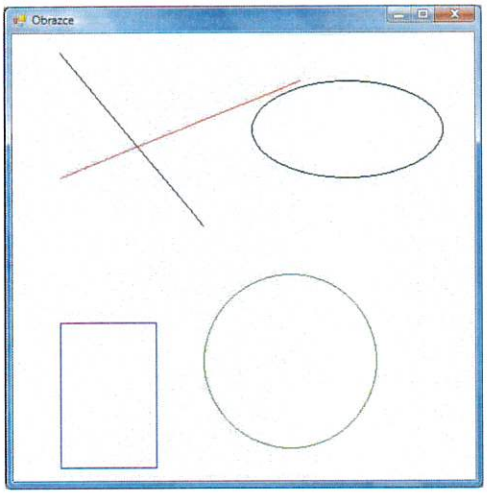
Př.

```
kp.DrawLine(Pens.Black,50,20,200,200);
```

Nakreslí černou úsečku z bodu [50,20] do bodu [200,200]

`kp.DrawRectangle(Pens.Blue,50,300,100,150);`
Nakreslí modrý obdélník s levým horním rohem [50,300], šířkou 100 a výškou 150. Čtverec je obdélník se stejnou šířkou a výškou.

`kp.DrawEllipse(Pens.Black,250,50,200,100);`
Nakreslí elipsu s levým horním rohem opsaného obdélníka [250,50], šířkou 200 a výškou 100. Kružnice je elipsa se stejnou šírkou a výškou



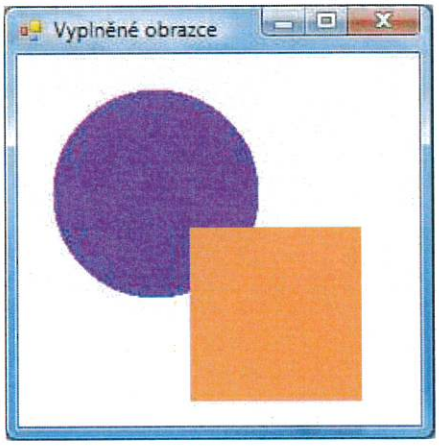

```
privatevoid OknoProgramu_Paint(object sender,
PaintEventArgs e)
{
  Graphics kp = e.Graphics;
  kp.DrawLine(Pens.Black,50,20,200,200);
  kp.DrawLine(Pens.Red,300,50,50,150);
  kp.DrawRectangle(Pens.Blue,50,300,100,150);
  kp.DrawEllipse(Pens.Black,250,50,200,100);
  kp.DrawEllipse(Pens.Green,200,250,180,180);
}
```

Vyplněné obrazce

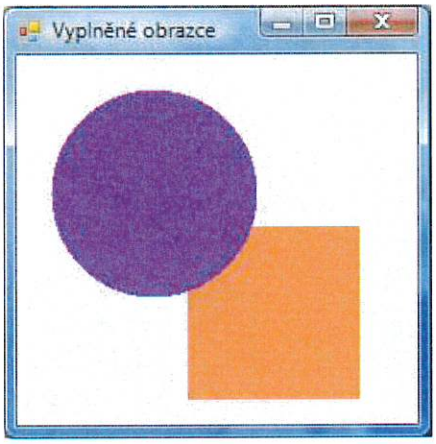
Kreslení vyplněných obrazců se od těch nevyplněných liší tím, že kreslíme štětcem (vybíráme ze „sady štětců“ `Brashes`) a názvy všech metod nezačínají `Draw`, ale `Fill`. Ostatní parametry jsou stejné
Př.


```
privatevoid OknoProgramu_Paint(object sender,
PaintEventArgs e)
{
Graphics kp = e.Graphics;
kp.FillEllipse(Brushes.BlueViolet,20,20,120,120)
;
kp.FillRectangle(Brushes.Coral,160,100,100,100);
}
```

Při překrývání obrazců bude navrch ten, který je nakreslený později




```
privatevoid OknoProgramu_Paint(object sender,
PaintEventArgs e)
{
   Graphics kp = e.Graphics;
   kp.FillEllipse(Brushes.BlueViolet,20,20,120,120);
   kp.FillRectangle(Brushes.Coral,100,100,100,100);
}
```

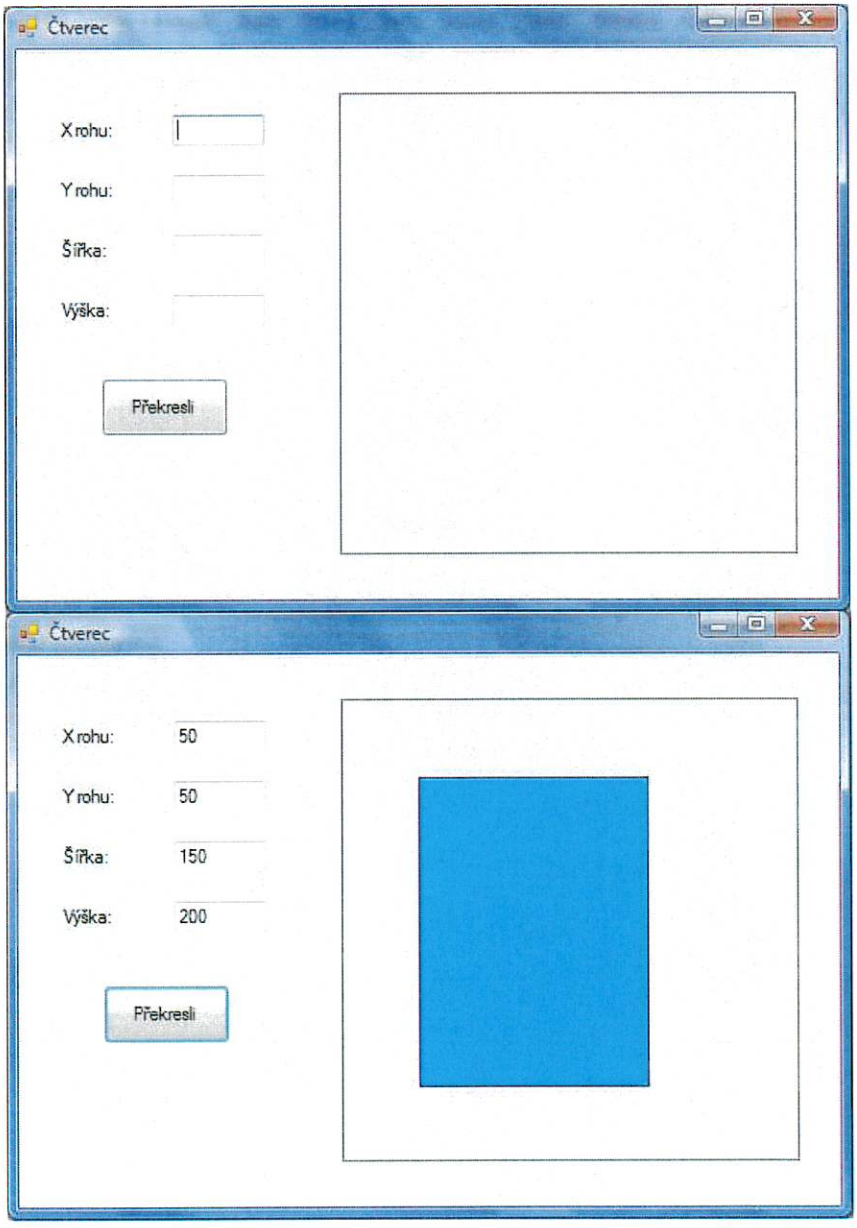



```
privatevoid OknoProgramu_Paint(object sender,
PaintEventArgs e)
{
    Graphics kp = e.Graphics;
    kp.FillRectangle(Brushes.Coral,100,100,100,100);
    kp.FillEllipse(Brushes.BlueViolet,20,20,120,120);
}
```

Panel

Pokud nechceme kreslit přímo do okna programu, zvolíme v Toolboxu nástroj „*panel*" ze skupiny *Containers*. Přetáhneme ho do okna programu a pro jeho ohraničení nastavíme vlastnost `BorderStyle` na hodnotu `FixedSingle`.

Pro kreslení do panelu napíšeme metodu `panel_click` . Používání metod pro kreslení graf. objektů je stejné, souřadný systém se bere vzhledem k panelu, takže bod [0,0] je v levém horním rohu panelu.



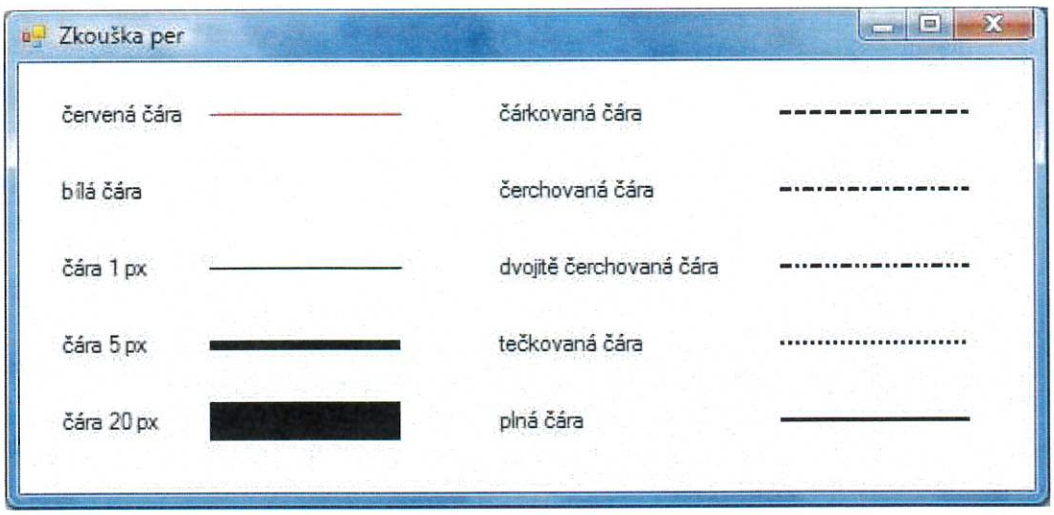
Připravíme si prostředí podle následujícího obrázku. Panelu nastavíme rozměry 300;300.
Po stisku tlačítka *Překresli* se do panelu vykreslí modrý (aqua) vyplněný obdélník s tmavě modrým okrajem (navy).

Souřadnice levého horního rohu, šířku a výšku vezmeme z textových polí. Hodnoty musíme převést z textového řetězce na celé číslo. Kvůli ošetření vstupních údajů napíšeme všechny převody do pokusného bloku `try`. Do větve catch napíšeme příkaz return; , který ukončí vykonávání obslužné metody. Tentokrát nemůžeme použít deklaraci s inicilizací, protože proměnné, které bychom deklarovali v bloku try, by mimo tento blok neexistovaly a my bychom je nemohli využít pro kreslení obdélníku.
Proměnné stejného typu můžeme deklarovat jedním příkazem:`int x, y, šířka, výška;`
Nejprve nakreslíme plný obdélník a pak teprve jeho obrys. Obráceně by se nám obrys překryl výplní.
Překreslení panelu vyvoláme příkazem `Panel.Refresh();`, který napíšeme do metody `TlačítkoPřekresli_Click`

```
privatevoid Panel_Paint(object sender, PaintEventArgs e)
{
   Graphics kp = e.Graphics;
   int x, y, šířka, výška;
   try
            {
                x = Convert.ToInt32(PoleX.Text);
                y = Convert.ToInt32(PoleY.Text);
                šířka = Convert.ToInt32(PoleŠířka.Text);
                výška = Convert.ToInt32(PoleVýška.Text);
            }
    catch
            {
              return;
            }

            kp.FillRectangle(Brushes.Aqua, x, y, šířka, výška);
            kp.DrawRectangle(Pens.Navy,x,y,šířka,výška);
 }

privatevoid TlačítkoPřekresli_Click(object sender, EventArgs e)
 {
            Panel.Refresh();
 }
```

Různá pera

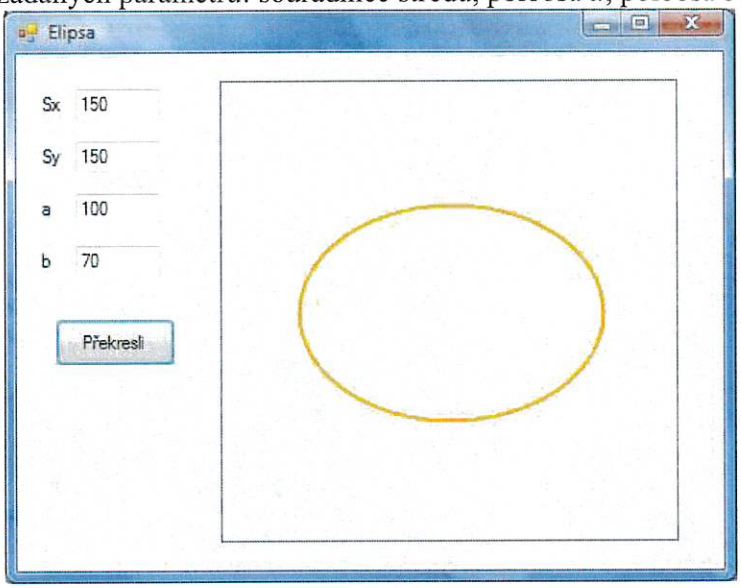
Pokud nám nestačí tenká plná čára ze sady per `Pens`, můžeme si vyrobit vlastní pero, jako objekt třídy `Pen`.
Pro vytvoření pera červené barvy napíšeme: `Pen MojePero=newPen(Color.Red);`
Jako druhý parametr můžeme uvést tloušťku pera. Není-li tloušťka uvedena, bere se 1px.
Barvu a tloušťku pera můžeme měnit zadáním hodnot do `MojePero.ColoraMojePero.Width.`
Pro definování stylu čáry musíme používat jmenný prostor `System.Drawing.Drawing2Da` na začátek souboru Form1.cs vložit řádek `using System.Drawing.Drawing2D;`
Styl čáry zadáme hodnotou do `MojePero.DashStyle`.
Hodnoty jsou: Dash – čárkovaná
Dot - tečkovaná
DashDot – čerchovaná (čárka-tečka)
DashDotDot – dvojitě čerchovaná (čárka-čárka-tečka)
Solid – plná

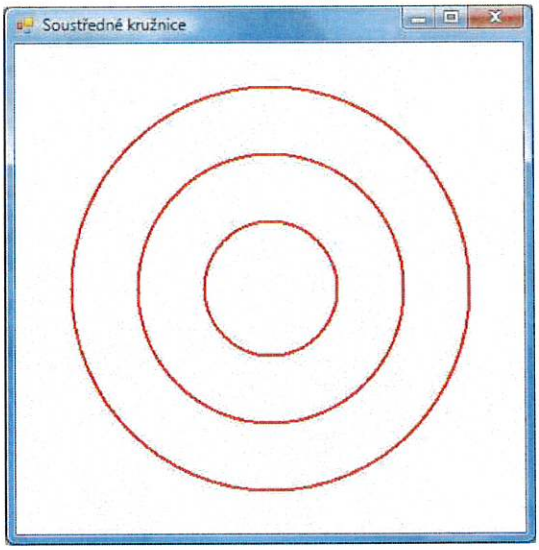



```
privatevoid OknoProgramu_Paint(object sender, PaintEventArgs e)
{
    Graphics kp = e.Graphics;
    Pen MojePero=newPen(Color.Red);
            kp.DrawLine(MojePero, 100,27,200,27);
            MojePero.Color = Color.White;
            kp.DrawLine(MojePero, 100, 67, 200, 67);
            MojePero.Color = Color.Black;
            MojePero.Width = 1;
            kp.DrawLine(MojePero, 100, 107, 200, 107);
            MojePero.Width = 5;
            kp.DrawLine(MojePero, 100, 147, 200, 147);
            MojePero.Width = 20;
            kp.DrawLine(MojePero, 100, 187, 200, 187);
            MojePero.Width = 2;
            MojePero.DashStyle = DashStyle.Dash;
            kp.DrawLine(MojePero, 400, 27, 500, 27);
            MojePero.DashStyle = DashStyle.DashDot;
            kp.DrawLine(MojePero, 400, 67, 500, 67);
            MojePero.DashStyle = DashStyle.DashDotDot;
            kp.DrawLine(MojePero, 400, 107, 500, 107);
            MojePero.DashStyle = DashStyle.Dot;
            kp.DrawLine(MojePero, 400, 147, 500, 147);
            MojePero.DashStyle = DashStyle.Solid;
            kp.DrawLine(MojePero, 400, 187, 500, 187);
    }
```

Příklady na procvičení:

Příklad 1: Napište program*Elipsa*, který do panelu nakreslí oranžovou elipsu s tloušťkou čáry 2px, podle zadaných parametrů: souřadnice středu, poloosa *a*, poloosa *b*.

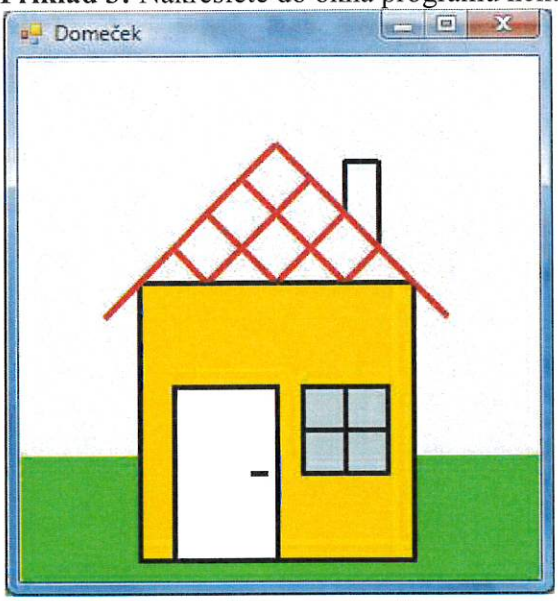


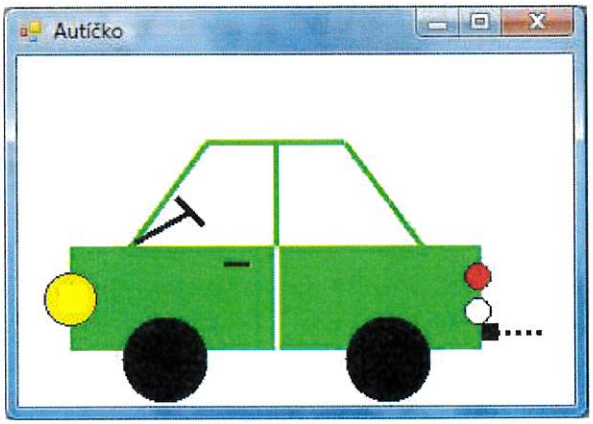


Příklad 2: Napište program *Soustředné kružnice*, který nakreslí do okna programu tři červené kružnice s tloušťkou čáry 2px, se středem v středu okna a poloměry 50, 100 a 150px.

Střed klientské části okna zjistíte:
```
int Sx = ClientSize.Width / 2;
int Sy = ClientSize.Height / 2;
```

Příklad 3: Nakreslete do okna programu některý z těchto obrázků: domeček, panáček, autíčko.





Práce s reálnými proměnnými

ČÍSELNÉ TYPY

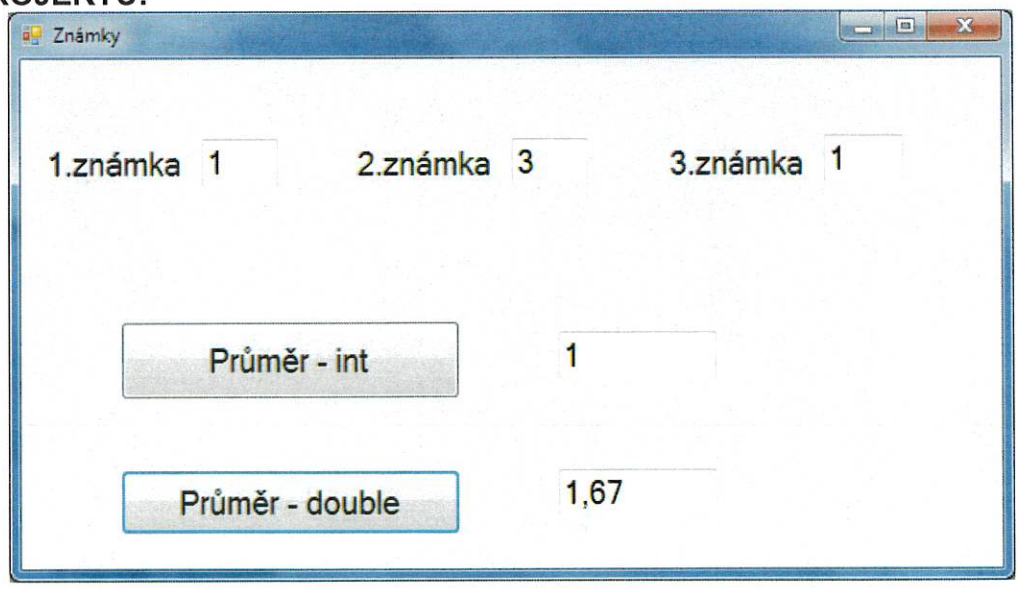
DRUH	NÁZEV TYPU	ALTERNAT. NÁZEV	ROZSAH HODNOT	PŘESNOST	POČET BITŮ
CELÁ ČÍSLA	byte	Byte	0 až 255		8
	sbyte	SByte	-128 až +127		8
	ushort	UInt16	0 až 65 535		16
	short	Int16	-32 768 až +32 767		16
	uint	UInt32	0 až 4 miliardy		32
	int	Int32	+- 2 miliardy		32
	ulong	UInt64	0 až 18 trilionů		64
	long	Int64	+- 9 trilionů		64
DESETINNÁ ČÍSLA	float	Single	+- 10^{38}	7 míst	32
	double	Double	+- 10^{308}	15 míst	64
DES.ČÍSLA S PŘESNOU REPREZENTACÍ	decimal	Decimal	+- 10^{28}	28 míst	128

Číselné hodnoty v aplikaci lze zadat do proměnných různých typů. Název typu určuje, jak velké číslo můžeme zadat, což je dáno velikostí (v bitech). Při používání více typů současně, platí tzv. implicitní typová konverze, kdy je možné hodnotu menšího typu přiřadit proměnné s typem s rozsahem větším. Tedy např. z typu **int** do typu **double**. Double ukládá číslo v exponenciálním tvaru ($8{,}5354*10^3$ = 8 535,4)

ŘEŠENÝ PŘÍKLAD

ZADÁNÍ:
Zadejte 3 známky. Vypočítejte průměr, porovnejte řešení s celočíselným a reálným typem.

NÁVRH PROJEKTU:



KÓD ŘEŠENÍ:

```
namespace prumerDouble
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            //získáme 3 zadané známky převedené na číslo
            int z1, z2, z3;
            z1 = Convert.ToInt32(poleZ1.Text);
            z2 = Convert.ToInt32(poleZ2.Text);
            z3 = Convert.ToInt32(poleZ3.Text);
            //vypočítáme celočíselný průměr
            int prumer = (z1 + z2 + z3) / 3;
            //převedeme průměr na text a zobrazíme
            polePrumer.Text =Convert.ToString(prumer);
        }
        private void button2_Click(object sender, EventArgs e)
        {
            //získáme 3 zadané známky převedené na číslo
            int z1, z2, z3;
            z1 = Convert.ToInt32(poleZ1.Text);
            z2 = Convert.ToInt32(poleZ2.Text);
            z3 = Convert.ToInt32(poleZ3.Text);
            //vypočítáme průměr
            double prumer = (z1 + z2 + z3) / 3;
            //převedeme průměr na text a zobrazíme s přesností na 2desetinná místa
            polePrumer2.Text = prumer.ToString("F2");
        }
    }
}
```

ZÁKLADNÍ PROBLÉM PŘI DĚLENÍ:

Pro vyjádření dělení používáme znak /, tento znak má ale dvojí význam. Buď pro celočíselné dělení nebo pro klasické. O jaké dělení skutečně půjde, závisí na obou členech dělení. Pokud jsou oba celočíselného typu, bude dělení celočíselné. Tedy např.:

0 = 1 / 4 ; 0 = 3 / 4 ; 1 = 5 / 4 ;

Jakmile je alespoň jeden z členů reálného typu, půjde o klasické dělení. Tedy např.:

0.25 = 1.0 / 4 ; 0.25 = 1 / 4.0 ; 0.75 = 3.0 / 4 ; 1.25 = 5.0 / 4.0 ;

V řešeném příkladu tak nejprve průměr, i když má nastavený typ **double**, bude opět celočíselný. Proto do zvýrazněné části kódu vložíme místo celého čísla 3 reálné číslo 3.0, viz:

```
double prumer = (z1 + z2 + z3) / 3.0;
```

Nebo známkám můžeme nadefinovat místo typu **int** typ **double**.

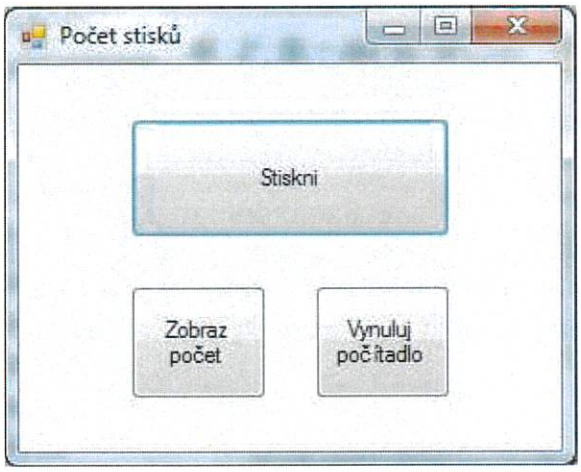
ČLENSKÉ PROMĚNNÉ

Při tvorbě aplikací často potřebujeme některé hodnoty používat v několika událostech, proto je musíme uložit do proměnné, která bude mít platnost pro celou aplikaci. Tyto proměnné se jmenují členské a deklarují se v části pro celý formulář. V paměti budou existovat tak dlouho, dokud poběží formulář.

ŘEŠENÝ PŘÍKLAD

ZADÁNÍ:
Počítejte počet stisků tlačítka Stiskni tak, aby se do proměnné Počet zapisovalo každé stisknutí, dokud nebude Počet vynulován. Tlačítko Zobraz počet ukáže okno se zprávou o počtu stisků.

NÁVRH PROJEKTU:



KÓD ŘEŠENÍ:

```
namespace počet_stisků
{
    public partial class Form1 : Form
    {
        //členská proměnná
        int počítadlo = 0 ;
        public Form1()
        {
            InitializeComponent();
        }

        private void bStisk_Click(object sender, EventArgs e)
        {
            //při každém stisku zvětši počet o 1
            počítadlo ++;
        }

        private void bVýmaz_Click(object sender, EventArgs e)
        {
            //vynuluj počet
            počítadlo = 0;
        }
        private void bPočet_Click(object sender, EventArgs e)
        {
            //vypiš počet
            MessageBox.Show("Počet je "+počítadlo.ToString());
        }
    }
```

}

OPERÁTORY A SLOŽENÉ PŘIŘAZENÍ

++ zvětši o 1 -- zmenši o 1 += 4 zvětši o 4 -= zmenši o 9 *= 2 zvětši 2krát

PROCVIČENÍ

Jaká bude hodnota A ?

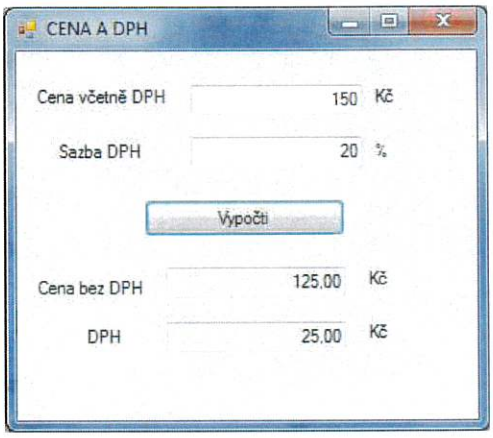
```
int  A = 8;
    A += 15;
    A *=  3;
    A--;
    A -= 7;
```

TYPOVÉ PŘÍKLADY:

1. Zadejte, váš rekord ve skoku do dálky v minulém roce a v tomto. O kolik procent jste se zlepšili?

2. Zadejte celkovou cenu výrobku a jaká je DPH v procentech. Po stisknutí tlačítka Zaplatit vypište cenu výrobku a kolik bude DPH.

3. Zadejte číslo, připravte dvě tlačítka. Jedním budete číslo zvětšovat o jedničku, druhým zmenšovat také o jedničku.



VĚTVENÍ PROGRAMŮ - IF

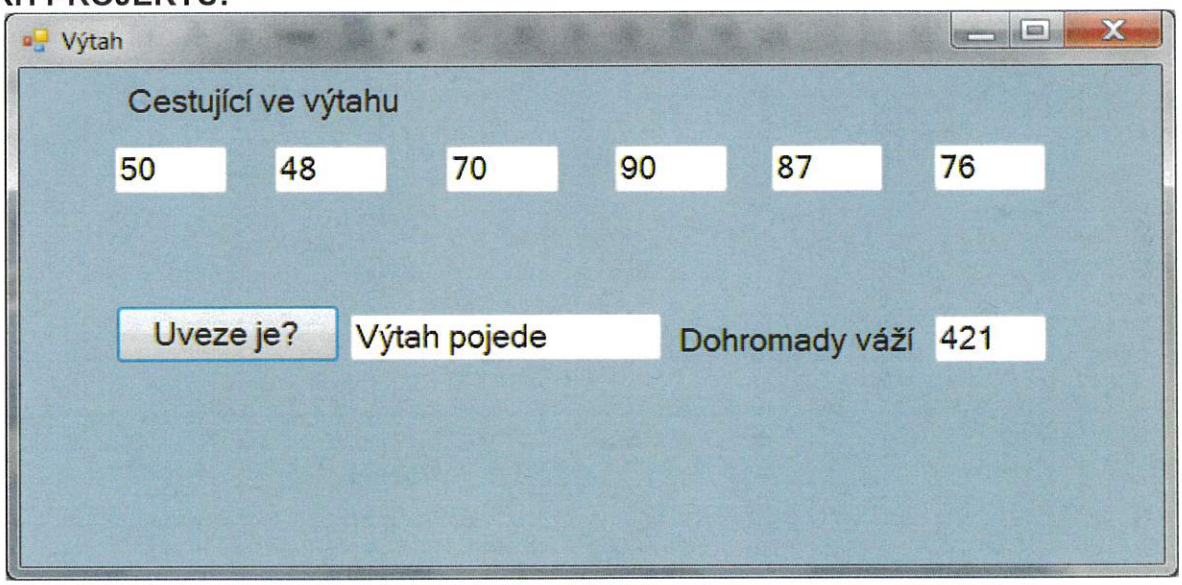
V programu často nastane situace, kdy další postup bude různý. Což obvykle závisí na splnění nějaké podmínky nebo na určité hodnotě řídící proměnné. V prvním případě použijeme příkaz IF. Jeho syntaxe:

```
if (podmínka )                    když podmínka platí
        příkaz 1;                 provede příkaz 1
else                              jinak (=když neplatí)
        příkaz 2;                 provede příkaz 2
```

ŘEŠENÝ PŘÍKLAD

ZADÁNÍ:
Výtah na internátě uveze 6 lidí, jeho nosnost je kg. Zadejte váhu 6 lidí. Vypočítejte, kolik váží dohromady a zda je výtah uveze.

NÁVRH PROJEKTU:



KÓD ŘEŠENÍ:

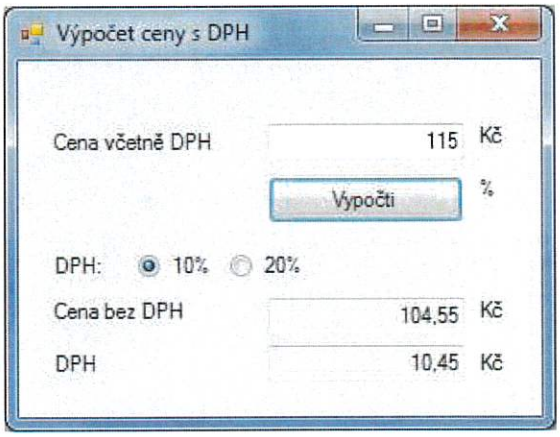
```
private void button1_Click(object sender, EventArgs e)
        {
            int v1, v2, v3, v4, v5, v6;
            //zjistíme váhu každého,vypočteme celkovou váhu
            //textboxy jsou přednastavené na nulu
            v1 = Convert.ToInt32(textBox1.Text);
            v2 = Convert.ToInt32(textBox2.Text);
            v3 = Convert.ToInt32(textBox3.Text);
            v4 = Convert.ToInt32(textBox4.Text);
            v5 = Convert.ToInt32(textBox5.Text);
            v6 = Convert.ToInt32(textBox6.Text);
            int celkem = v1 + v2 + v3 + v4 + v5 + v6;
            poleVaha.Text = celkem.ToString();
            //druhá možnost výpisu
            //poleVaha.Text = Convert.ToString(celkem);
```

```
            //porovnáme celkovou hmotnost s nosností
            if (celkem <= 600)
                poleUveze.Text = "Výtah pojede";
            else
                poleUveze.Text = "Výtah nepojede";
            }
```

ŘEŠENÝ PŘÍKLAD

ZADÁNÍ:
Spočítejte podle zadané ceny a sazby DPH, cenu bez daně a samotnou DPH.

NÁVRH PROJEKTU:



KÓD ŘEŠENÍ:

```
namespace výpočet_ceny
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void bVýpocet_Click(object sender, EventArgs e)
        {
                double cena = Convert.ToDouble(tCenaDPH.Text);
                double sazba;
                //nastavení sazby daně
                if (rad10.Checked)
                 sazba = 10;
                else
                 sazba = 20;
                //výpočet ceny a DPH
                double cenaBezDPH = cena / (1+ sazba / 100);
                double dph = cena - cenaBezDPH;
                //výpis výpočtu na 2desetinná místa
                tDPHCen.Text = dph.ToString("F2");
                tCenaBezDPH.Text = cenaBezDPH.ToString("F2");          }
    }
}
```

ODBORNĚJŠÍ ZÁPIS

```
//pokud je v textboxu zadáno reálné číslo
if (this.isDouble(tCenaDPH.Text) )
{
    double cena = Convert.ToDouble(tCenaDPH.Text);
    //nastavení sazby podle zaškrtnutého radio tlačítka
    double sazba = (rad10.Checked ? 10:20);
    double cenaBezDPH = cena / (1+ sazba / 100);
    double dph = cena - cenaBezDPH;

    tDPHCen.Text = dph.ToString("F2");
    tCenaBezDPH.Text = cenaBezDPH.ToString("F2");
  }
```

LOGICKÉ OPERÁTORY A SPOJOVÁNÍ PODMÍNEK

== rovná se **<>** nerovná se **<=** menší nebo rovno **>=** větší nebo rovno

AND – a zároveň &&
OR – nebo ||
NEGACE – opak !
(A >= 3) && (A <= 25) A je z intervalu od 3 do 25
(heslo==“admin“) || (heslo==“a4“) heslo je admin nebo a4

VĚTVENÍ PROGRAMŮ - SWITCH

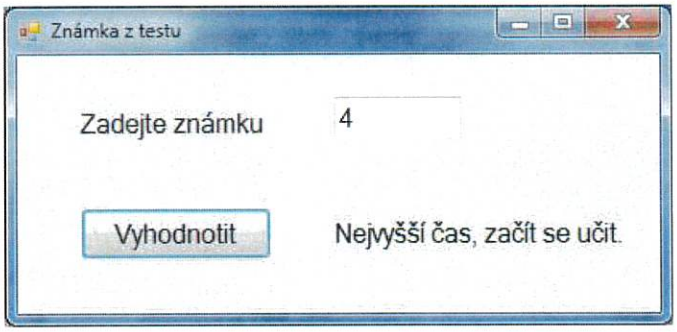
V případě, že postup programu závisí na určité hodnotě řídící proměnné, použijeme příkaz SWITCH - přepínač. Jeho syntaxe:

```
switch (jméno proměnné)
    {       case hodnota1:
              příkaz 1;
              break;
            .....
            case hodnota N:
              příkaz N;
              break;
            default:
              příkaz pokud to není ani jedna ze zadaných hodnot;
              break;
     }
```

ŘEŠENÝ PŘÍKLAD

ZADÁNÍ:
Zadejte vaši známku z testu a vypište vyhodnocení.

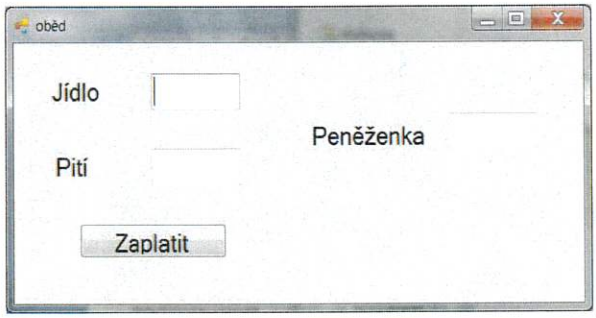
NÁVRH PROJEKTU:



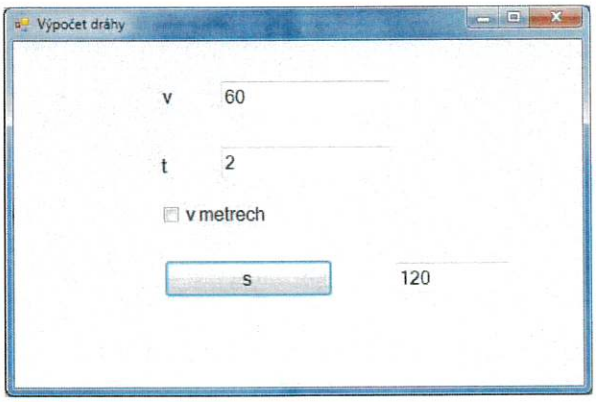
KÓD ŘEŠENÍ:

```
{   int znamka = Convert.ToInt32(poleZnamka.Text);
//podle zadané známky se zobrazí hodnocení
            switch (znamka)
                {
                    case 1:
                        výpisHodnocení.Text = "Je to super.";
                        break;
                    case 2:
                        výpisHodnocení.Text = "Jde to.";
                        break;
                    case 3:
                        výpisHodnocení.Text = "Nic moc.";
                        break;
                    case 4:
                        výpisHodnocení.Text = "Nejvyšší čas, začít se učit.";
                        break;
                    case 5:
                        výpisHodnocení.Text = "Není už pozdě?";
                        break;
                    default:
                        výpisHodnocení.Text = "Špatně zadaná známka.";
                        break;
                }
```

TYPOVÉ PŘÍKLADY:



1. Zadejte, kolik stojí jídlo a pití, které si chcete dát k obědu a kolik peněz máta v peněžence. Stačí vám to?



2. Zadejte rychlost a čas, vypočítejte dráhu, zvolte její jednotky.

3. Zadejte cenu jízdenky, počet osob. Vypočtěte celkovou cenu, když dodržíte pravidlo Českých drah: první osoba jede za plnou cenu, každý další platí poloviční cenu.

Další práce s reálnými proměnnými - funkce

DRUH	NÁZEV TYPU	ALTERNAT. NÁZEV	ROZSAH HODNOT	PŘESNOST	POČET BITŮ
CELÁ ČÍSLA	byte	Byte	0 až 255		8
	sbyte	SByte	-128 až +127		8
	ushort	UInt16	0 až 65 535		16
	short	Int16	-32 768 až +32 767		16
	uint	UInt32	0 až 4 miliardy		32
	int	Int32	+- 2 miliardy		32
	ulong	UInt64	0 až 18 trilionů		64
	long	Int64	+- 9 trilionů		64
DESETINNÁ ČÍSLA	float	Single	+- 10^{38}	7 míst	32
	double	Double	+- 10^{308}	15 míst	64
DES.ČÍSLA S PŘESNOU REPREZENTACÍ	decimal	Decimal	+- 10^{28}	28 míst	128

Platí tzv. **implicitní typová konverze**, kdy je možné hodnotu menšího typu přiřadit proměnné s typem s rozsahem větším. Tedy např. z typu **int** do typu **double**.
Lze také použít **explicitní typovou konverzi**, je možné hodnotu většího typu přiřadit proměnné s typem s menším rozsahem. Tedy např. z typu **double** do typu **int**. Pozor, tento typ konverze ale odřízne desetinnou část! V proměnné ***celé*** bude číslo 12.

```
int celé = (int) 12.8 ;
```

ZÁKLADNÍ FUNKCE PRO PRÁCI S REÁLNÝMI PROMĚNNÝMI

Základní aritmetické operace se zapisují jako v matematice, ale tam někdy vynecháváme násobení např. 3B, kdy v aplikaci musíme napsat 3*B. Také zde má násobení a dělení přednost před sčítáním a odčítáním. Proto pokud potřebujeme výraz vyhodnotit v jiném pořadí, musíme použít kulaté závorky. Pro práci s reálnými proměnnými můžeme používat matematické funkce – metody třídy Math.

Mocnina	$y = 5^{x-3}$	y = Math.Pow(5, x-3);
Druhá odmocnina	$y = \sqrt{x}$	y = Math.Sqrt(x);
Absolutní hodnota	$y = \lvert x-8 \rvert$	y = Math.Abs(x-8);
Logaritmus	$y = \log_2 (x+5)$	y = Math.Log(x+5, 2);
Zaokrouhlení na 3 desetinná místa		y = Math.Round(x , 3);

Goniometrické funkce (**pouze** pro úhly v radiánech)
Přepočet z úhlů na radiány: uhelRad = Math.PI / 180 * uhelStupne;

Sinus	y = sin 2x	y = Math.Sin(2*x);
Cosinus	y = cos x	y = Math.Cos(x);
Tangens	y = tg 3x	y = Math.Tan(3*x);

PROCVIČENÍ
Přepište matematické výrazy do jazyka c#?

$$z = \sin(3x) + \cos(2y)$$

$$y = 5x - 18/\ 2x\ - 20$$

$$y = 2.5\ .\ 10^{3x}$$

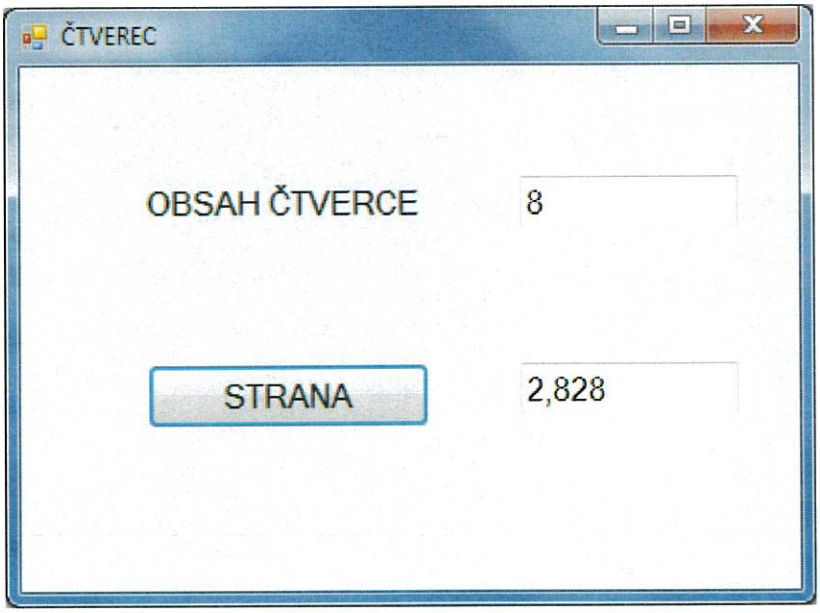
LOGICKÉ OPERÁTORY

==	ROVNÁ SE
!	NEGACE
&&	AND
||	OR

ŘEŠENÝ PŘÍKLAD

ZADÁNÍ:
Zadejte obsah čtverce. Vypočítejte velikost jedné strany.

NÁVRH PROJEKTU:



```
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            //připravíme dvě reálné proměnné
            double obsah, stranaA;
            //načteme a převedeme obsah
            obsah = Convert.ToDouble(poleObsah.Text);
            //použijeme funkci pro druhou odmocninu
            stranaA = Math.Sqrt(obsah);
            //zobrazíme vypočítanou stranu na 3 desetinná místa
            poleStrana.Text = stranaA.ToString("F3");

        }
    }
}
```

TYPOVÉ PŘÍKLADY:

1. Vypočítejte výšku trojúhelníka ze zadaných tří stran, vypište výsledek.

2. Zadejte úhel ve stupních, vypište jeho goniometrické funkce. Nezapomeňte ho převést na radiány.

3. Zadejte vstupní údaje a vypočítejte výraz: $y = | \log_2 (x-8) |$

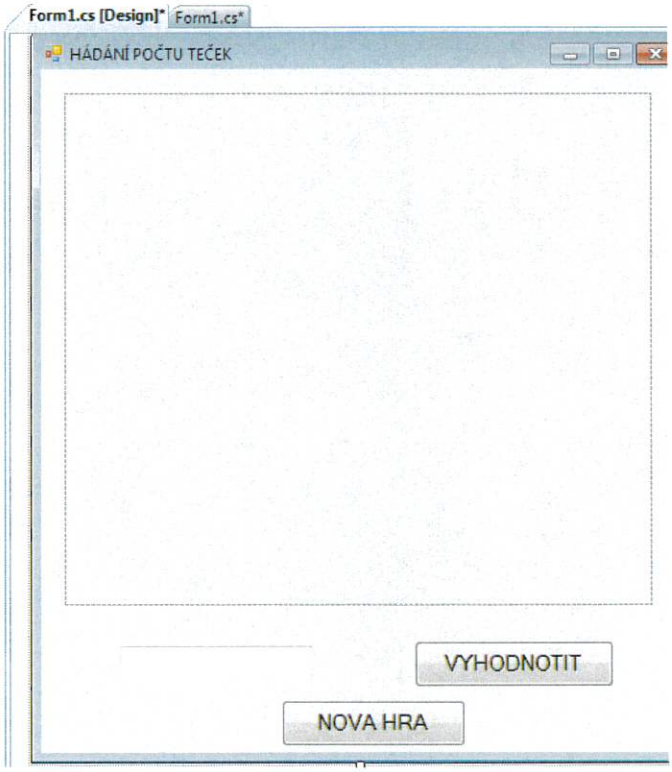
4. Zadejte vstupní údaje a vypočítejte, kolik bude na účtě po N letech při úroku P podle vzorce: částka = vklad * $(1 + P/100)^N$

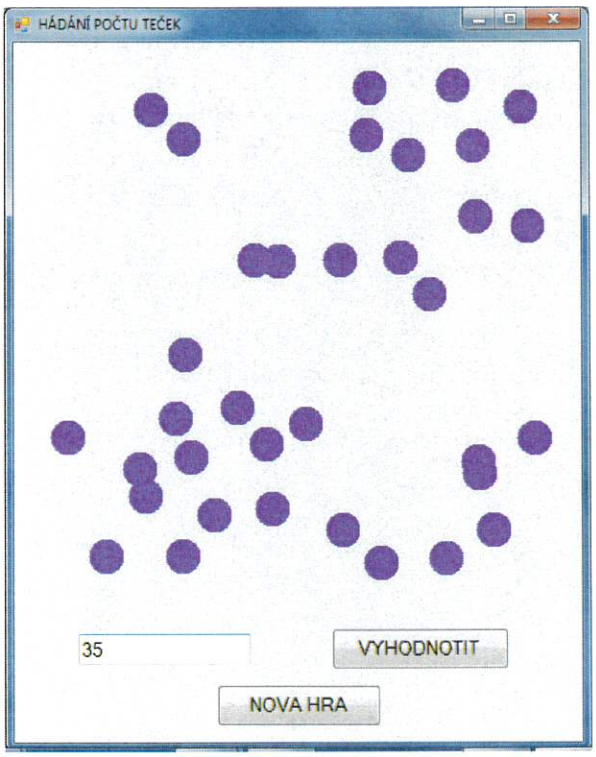
Generátor náhodných čísel

ŘEŠENÝ PŘÍKLAD

ZADÁNÍ:
Zadejte.

NÁVRH PROJEKTU:





KÓD ŘEŠENÍ:

```
public partial class Form1 : Form
    {
        // členské proměnné - generátor náhodných čísel,
        // náhodný počet teček, počet pokusů

        Random náhoda = new Random();
        int pocet, pokus=0;

        public Form1()
        {
            InitializeComponent();
        }

        private void panel1_Paint(object sender, PaintEventArgs e)
        {
            Graphics kp = e.Graphics;

          // metoda Next losuje počet teček od 10 do 50
             pocet = náhoda.Next(10, 51);

          // cyklus proběhne tolikrát, kolik je počet teček
          // pro vykreslení každé tečky je třeba vylosovat
          // náhodné souřadnice x, y

            for (int i = 1; i <= pocet; i++)
```

```
        {
            int x = náhoda.Next(0, 420);
            int y = náhoda.Next(0, 420);
            kp.FillEllipse(Brushes.BlueViolet , x, y, 30, 30);
        }
    }

    private void button1_Click(object sender, EventArgs e)
    {
        // vykreslení nových teček
        panel1.Refresh();
        // vynulování počtu pokusů a odhadovaného počtu teček
        poleOdhadTecek.Text = null;
        pokus = 0;

    }

    private void button2_Click(object sender, EventArgs e)
    {
        int zadáno = 0;
        //přičtení nového pokusu
        pokus++;
        // zjištění a převod odhadovaného počtu teček
        try
        {
            zadáno = Convert.ToInt32(poleOdhadTecek.Text);
        }
        catch { MessageBox.Show("Zadejte číslo"); }
        // porovnání skutečného a odhadovaného počtu teček
        if (pocet == zadáno)
          MessageBox.Show("Uhodli jste!Počet pokusů: "
              +pokus.ToString());
        else
        //  místo pro vložení vylepšení hry!

            if (pocet > zadáno)
              MessageBox.Show("Počet je větší");
            else
              MessageBox.Show("Počet je menší");

    }
```

VYLEPŠENÍ HRY:

Při nesprávném 10. pokusu ukončíme hru a znepřístupníme tlačítko Vyhodnotit, dokud nebude spuštěna Nová hra.

```
        if (pokus==10)
            {
              MessageBox.Show("Počet pokusů byl vyčerpán");
              buttonVyhodnotit.Enabled = false;
            }
        else
```

Do obsluhy tlačítka Nové hry vložíme zpřístupnění tlačítka Vyhodnotit.

```
              buttonVyhodnotit.Enabled = true;
```

TYPOVÉ PŘÍKLADY:

1. Vykreslete 5 náhodně umístěných čar.

2. Vypočítejte kvadratickou rovnici pro náhodně vylosovaná a, b, c. Rozsah intervalu pro ně zadejte (pomocí textBoxu), vypište výsledek.

CYKLUS S DANÝM POČTEM OPAKOVÁNÍ - FOR

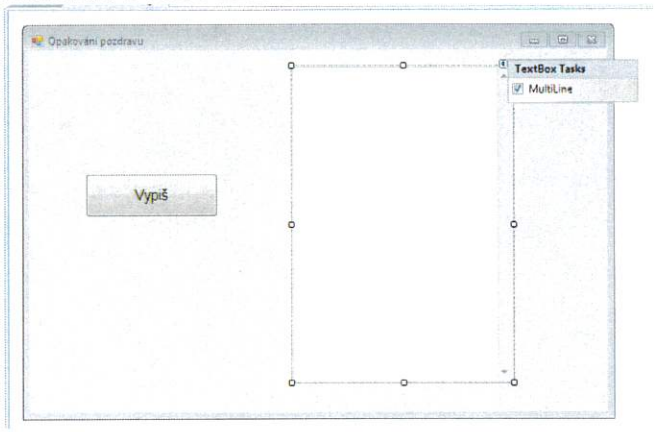
V programu občas nastane situace, kdy potřebujeme část kódu několikrát zopakovat. Pro počet opakování nastavíme celočíselnou řídící proměnnou. Syntaxe příkazu FOR:

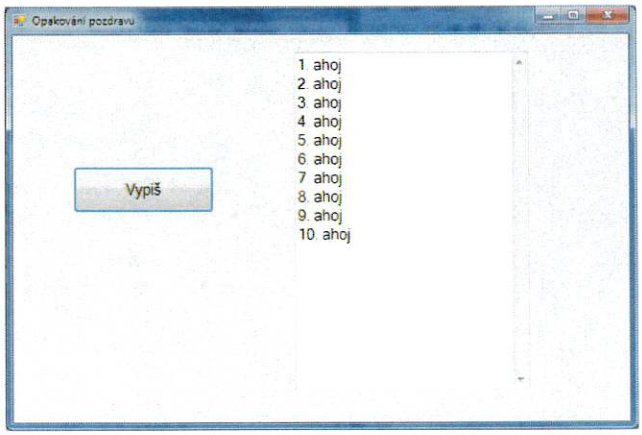
for (výchozí hodnota; konečná hodnota; krok-změna hodnoty) *opakovat, dokud*
příkaz 1; *nedosáhne konečné hodnoty, provede příkaz 1*

ŘEŠENÝ PŘÍKLAD 1

ZADÁNÍ:
Vypište desetkrát slovo AHOJ. Použijte pro výpis Textbox, kterému nastavíte vlastnost Multiline.

NÁVRH PROJEKTU:





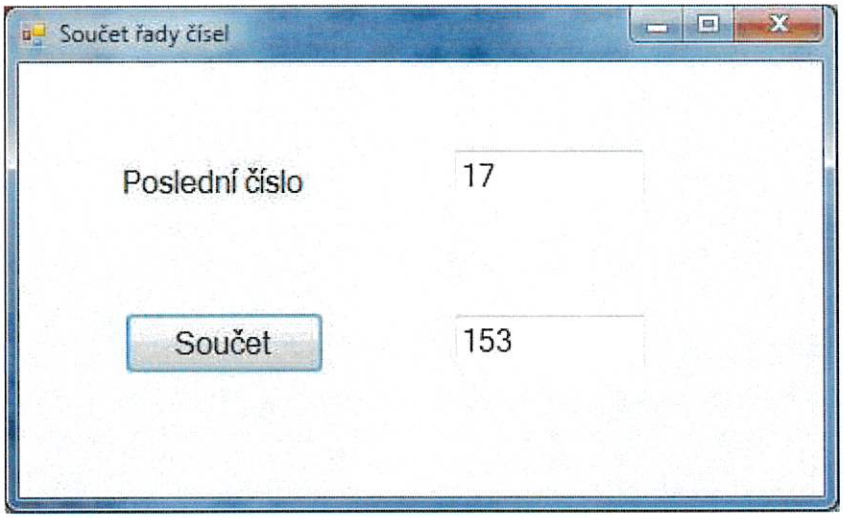
KÓD ŘEŠENÍ:

```
private void button1_Click(object sender, EventArgs e)
        {
            //nastavení opakování cyklu od 1 do 10 s krokem jedna(tedy +1)
            for (int p = 1; p <= 10 ; p++ )

            // přidání dalšího řádku, proto je použito +=
            // p je pořadí pozdravu, NewLine - přechod na nový řádek
            textBox1.Text += p.ToString()+ ". ahoj"+ Environment.NewLine;

            // jednoduchý výpis - pouze slova ahoj:
            // textBox1.Text += "ahoj" + Environment.NewLine;
        }
```

ŘEŠENÝ PŘÍKLAD 2

ZADÁNÍ:
Spočítejte součet řady čísel od 1 do zadaného posledního čísla.

NÁVRH PROJEKTU:



KÓD ŘEŠENÍ:

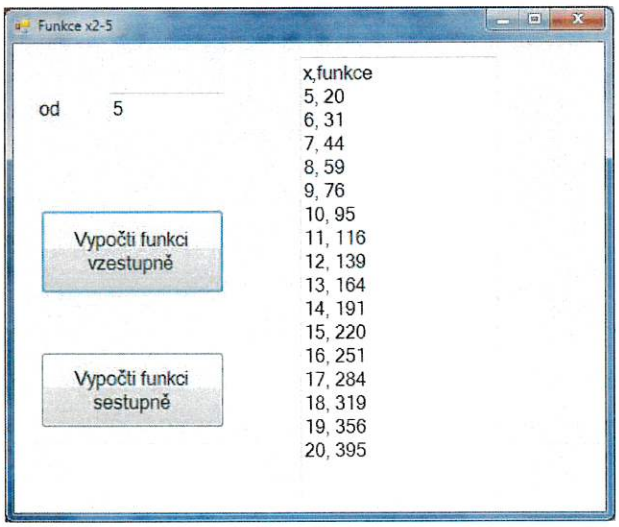
```
private void button1_Click(object sender, EventArgs e)
    {
     //zjistíme poslední číslo řady
     int číslo = Convert.ToInt32(textBox1.Text);
     //nadeklarujeme proměnnou pro součet a nastavíme výchozí hodnotu
     int s = 0;
     //v cyklu nemusí být přímo zadané hodnoty, ale mohou být zadány v proměnných
     for (int p = 1; p <= číslo; p++)
          //součet zvětšíme o jedno číslo řady
          s += p;
     //výpis součtu
     textBox2.Text = s.ToString();
    }
```

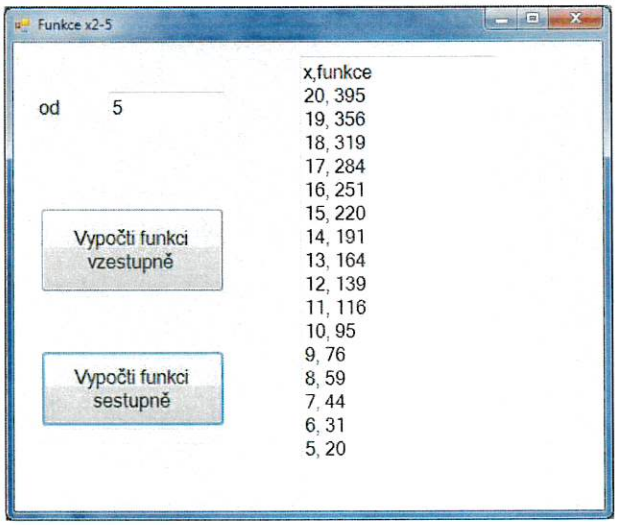
ŘEŠENÝ PŘÍKLAD 3

ZADÁNÍ:

Vypočítejte a vypište hodnoty funkce $f(x) = x^2 - 5$. Zadejte dolní hranici pro x, horní hranice bude 20. Hodnoty funkce vypište nejen vzestupně, ale i sestupně.

NÁVRH PROJEKTU:





KÓD ŘEŠENÍ:

```
private void button1_Click(object sender, EventArgs e)
    {
     //příprava prvního řádku
     poleVýpis.Text = "x,funkce" + Environment.NewLine;

     //nastavení dolní hranice pro výpočet funkce
     int od = Convert.ToInt32(textBox1.Text);

     //příprava proměnné pro funkci
     int funkce = 0;

     //výpočet funkce od zadaného x do x = 20
     for (int x = od; x <= 20; x++)
        {
         funkce = x * x -5;
         //přidání další řádky do výpisu, proto je použito +=
         poleVýpis.Text += x.ToString()+", "+funkce.ToString()+ Environment.NewLine;
        }
    }

 private void button2_Click(object sender, EventArgs e)
    {
     //příprava prvního řádku
     poleVýpis.Text = "x,funkce" + Environment.NewLine;

     //nastavení dolní hranice pro výpočet funkce
     int od = Convert.ToInt32(textBox1.Text);

     //příprava proměnné pro funkci
     int funkce = 0;

     //výpočet funkce od x = 20 do zadaného x, kde x se snižuje o 1
     for (int x = 20; x >= od; x--)
        {
         funkce = x * x - 5;
         //přidání další řádky do výpisu, proto je použito +=
        poleVýpis.Text += x.ToString()+ ", "+ funkce.ToString()+Environment.NewLine;
        }
```

TYPOVÉ PŘÍKLADY:

1. Zadejte, pro které číslo (od 1 do 10) chcete vypsat malou násobilku.

2. S použitím cyklu for, vypište všechna sudá čísla do 100. Nezapomeňte zapnout posuvník – Scrollbar pro Textbox.

3. Zakreslete 10 soustředných kružnic. Využijte cyklus pro zvětšování poloměru a určení souřadnic levého horního rohu.

POLE

Pole je datová struktura složená z hodnot stejného typu, které se rozlišují pomocí indexu. Takže ho používáme v aplikacích, kdy potřebujeme uložit a zpracovávat větší množství údajů stejného typu (např. známky, platy, počty vyrobených kusů …). Pro vytvoření této „hromadné" proměnné stačí jeden příkaz, všechny údaje lze zpracovávat jednotně pomocí cyklu a ke každé hodnotě lze přistupovat pomocí jejího pořadí v poli.
Vytvoříme tak např. pole pro zadání maximálně 10 známek:

```
int[] Známky = new int[10];
```

Známky

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

Známku do první buňky můžeme zadat:

```
Známky[0] = 1;
```

Pro výpočet průměru si připravíme proměnnou Součet, do které postupně přičteme všechny známky z pole pomocí cyklu For.

```
double Součet = 0;
for ( int pořadí = 0 ; pořadí < početHodnotVPoli; pořadí++)
    Součet += Známky[pořadí];
```

A pak už jen vypočítáme průměr a zobrazíme:

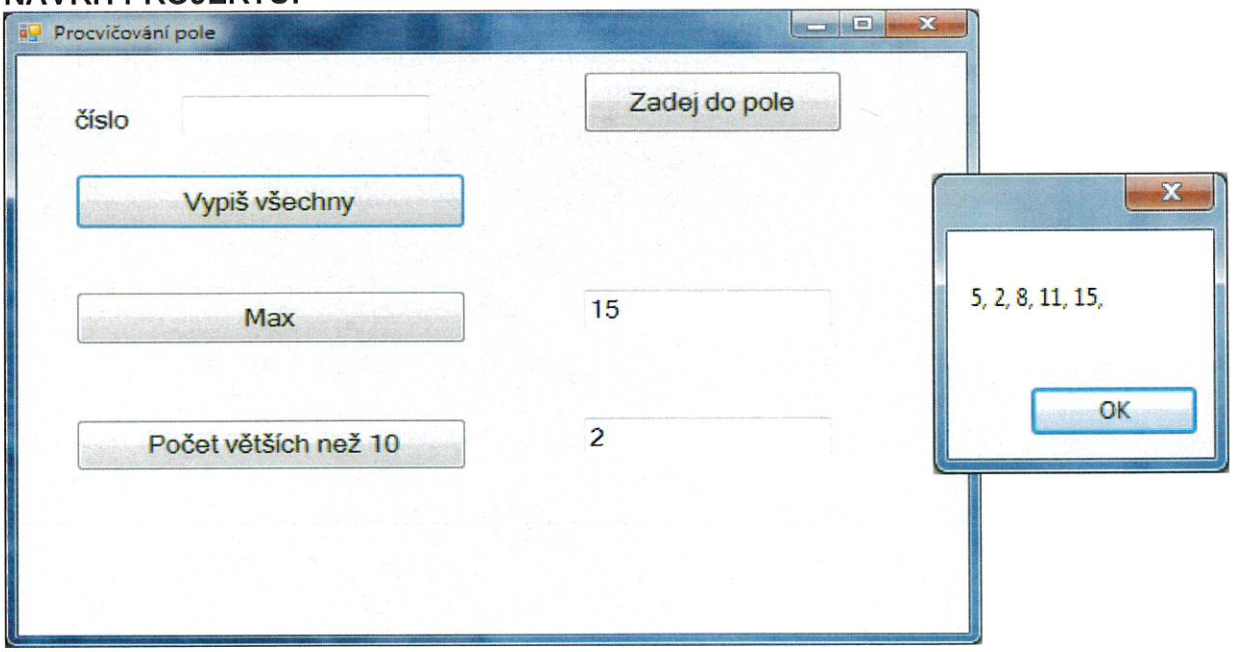
```
double průměr = Součet/ početHodnotVPoli;
```

ŘEŠENÝ PŘÍKLAD

ZADÁNÍ:

Připravte aplikaci pro zadání hodnot do číselného pole, zobrazení všech zadaných hodnot. Dále mezi zadanými hodnotami najděte největší a zjistěte, kolik z nich splňuje zadanou podmínku (jsou větší než 10).

NÁVRH PROJEKTU:



KÓD ŘEŠENÍ:

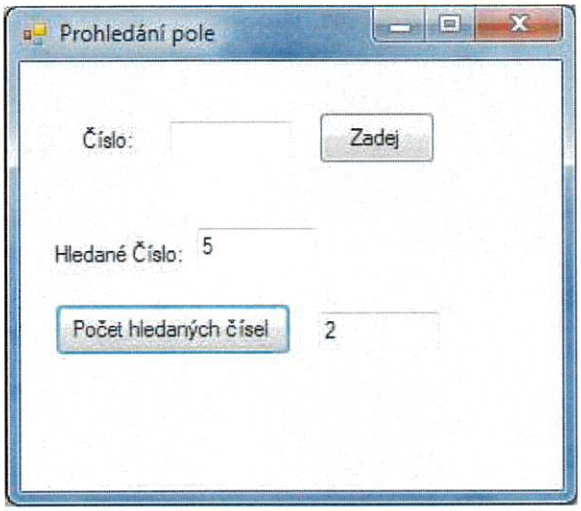
```
namespace poleNaTest
{
    public partial class Form1 : Form
    {
      // Členské proměnné
        int[] čísla = new int[10];
        int index = 0;

        public Form1()
        {
            InitializeComponent();
        }

        private void zadej_Click(object sender, EventArgs e)
        {
      // Uložení čísla do pole na pořadí INDEX
            čísla[index] = Convert.ToInt32(poleCislo.Text);

      // posunutí pořadí na další buňku
            index++;

      // vymazání textboxu
            poleCislo.Text = null;
        }

        private void vypis_Click(object sender, EventArgs e)
        {
            string zprava = null;
      // přidání do zprávy každého čísla z pole
            for (int i = 0; i < index; i++)
                zprava += čísla[i].ToString() + ", ";

      // zobrazení zprávy
            MessageBox.Show(zprava);
        }

        private void maximum_Click(object sender, EventArgs e)
        {
      // Vyhledání největšího čísla v poli, zvolíme nulté za největší
            int max = čísla[0];

      // prohledání pole od dalšího čísla a porovnání s dočasně největším
      // když je větší, přepíšeme ho novou hodnotou
            for (int i = 1; i < index; i++)
                if (max < čísla[i])
                    max = čísla[i];

      // zobrazení maxima
            poleMax.Text = max.ToString();
        }



        private void vetsi_Click(object sender, EventArgs e)
        {
       // proměnná pro počet
            int p = 0;
```

```
        // prohledání pole od začátku a porovnání se zadaným číslem
        // když je větší, zvětšíme počet p ojeden kus
             for (int i = 0; i < index; i++)
                  if (čísla[i] > 10)
                       p++;
         // zobrazení počtu
              polePocet.Text = p.ToString();
          }
    }
}
```

ZADÁNÍ:

Připravte aplikaci pro zadání hodnot do číselného pole. Dále zadejte hodnotu hledaného čísla mezi zadanými hodnotami, zjistěte, kolik jich je.

NÁVRH PROJEKTU:



KÓD ŘEŠENÍ:

```
namespace Pole_hledani
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        // Členské proměnné
        int[] čísla = new int[10];
        int index = 0;
        // Uložení čísla do pole
        private void tl_zadej_Click(object sender, EventArgs e)
        {
         // kontrola, jestli index není větší než délka pole
            if (index >= čísla.Length)
            {
                MessageBox.Show("Pole naplněno");
                return;
            }
            try
```

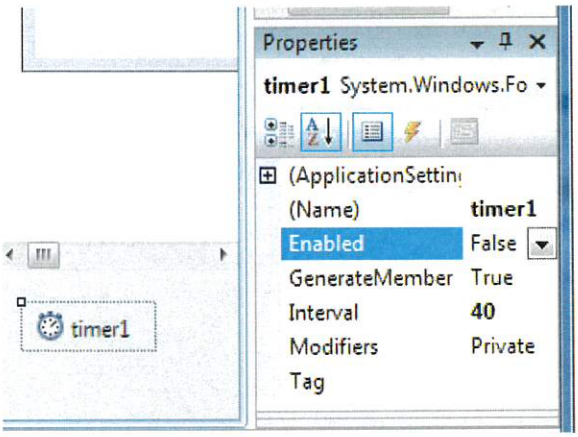
```
            {
                int číslo = Convert.ToInt32(pole_číslo.Text);
                čísla[index] = číslo;
                index++;
                pole_číslo.Text = null;
            }
            catch{}
        }
        // Vyhledání čísel v poli
        private void tl_hledej_Click(object sender, EventArgs e)
        {
            int počet = 0;
            bool nenalezeno = true;
            int hledané_číslo = Convert.ToInt32(pole_hledané_číslo.Text);
            for (int i = 0; i < index; i++)                {
                if (čísla[i] == hledané_číslo)
                {
                    počet++;
                    nenalezeno = false;
                }
            }
            if (!nenalezeno)
             pole_počet_čísel.Text = počet.ToString();
            else
                MessageBox.Show("Číslo nenalezeno");
        }
    }
}
```

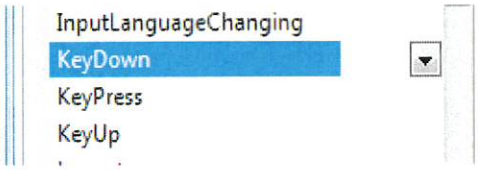
DALŠÍ PŘÍKLADY:

1. Vaše firma má nejvíce 12 zaměstnanců, zadejte loňské hodnoty prémií, pro letošní rok je zvyšte o 15 procent. Nové hodnoty zobrazte.
2. Do pole Balíky zadejte váhu všech balíků. Jakou nosnost musí mít dodávka, která je bude doručovat, vypište ji.
3. Zadejte číslo, o které snížíte všechny hodnoty v poli Čísla. Nové hodnoty zobrazte.

ANIMACE

V programu občas nastane situace, kdy potřebujeme obrázek rozpohybovat, tedy posunout ho, změnit jeho souřadnice. Pro lidské oko bude pohyb plynulý, když se obraz změní 25krát za sekundu (jako u filmového pásu).
Pro tuto změnu potřebujeme nějaký spínač, který sepne každou 0,04 s, tedy použijeme časovač – timer a jeho interval nastavíme na 40ms. Jeho další důležitou vlastností je Enabled, která spouští a vypíná spínání = tikání. Timer má jedinou událost Tick, kdy sepne. V této události potřebujeme změnit x-ovou souřadnici obrázku a znovu obrázek vykreslit na nové pozici, tedy obnovit – Refresh.





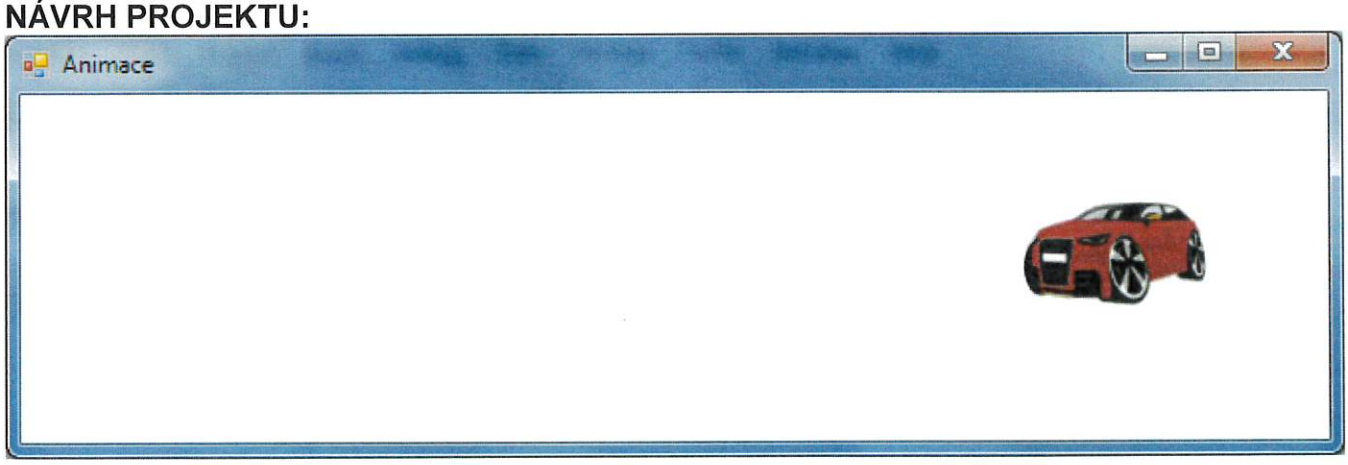
Na ovládání formuláře pomocí myši a klávesnice slouží události se slovy Key nebo Mouse v názvu, sloveso v názvu určuje jako je klávesa nebo tlačítko stisknuté.

Dále můžeme aplikaci vylepšit, aby obraz neproblikával, tím, že pro celý formulář nastavíme vlastnost DoubleBuffered na true, tedy zdvojíme paměť, v první je zobrazené auto a v druhé se připravuje obrázek na nových pozicích a vykreslí se, až bude připraven celý.

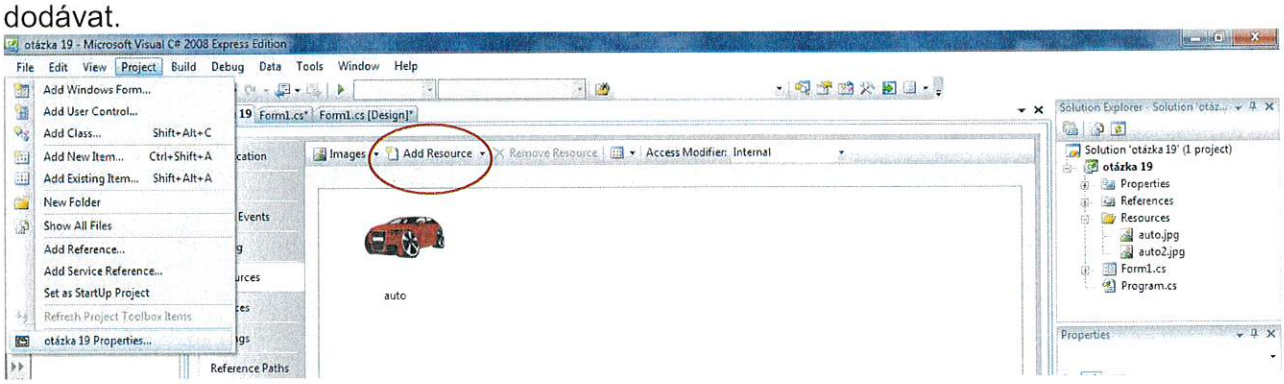
ŘEŠENÝ PŘÍKLAD

ZADÁNÍ:
Vykreslete na formulář obrázek auta, ovládejte ho pomocí klávesnice - šipky vlevo a vpravo. Šipka vlevo – auto se rozjede. Šipka vpravo – auto zastaví.

NÁVRH PROJEKTU:



Obrázek auta vložíme přes nabídku Project – Properties do jeho zdrojů (Resources) pomocí Add resources – add existing, stane se tak součástí souboru .exe a nemusíme ho k aplikaci dodávat.



KÓD ŘEŠENÍ:

```
public partial class OknoProgramu : Form
    {
        //členské proměnné: souřadnice x auta, obrázek auta
        int xAuta = 500;
        Image obrázek = Properties.Resources.auto;

        public OknoProgramu()
        {
            InitializeComponent();
        }

        private void OknoProgramu_Paint(object sender, PaintEventArgs e)
        {
            //vykreslení obrázku auta
            Graphics kp = e.Graphics;
            kp.DrawImage(obrázek,xAuta,50,100,63);
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            //posun auta doleva - k nule o 5bodů
            xAuta -= 5;
            //obnova - vykreslení na nových pozicích obrázku
            Refresh();
        }

        private void OknoProgramu_KeyDown(object sender, KeyEventArgs e)
        {
            // e.KeyCode je kód klávesy, která vyvolala událost stisknutá klávesa
            //klávesa šipka vlevo zapne spínač - auto se rozjede
            if (e.KeyCode == Keys.Left)
                timer1.Enabled = true;
            //klávesa šipka vpravo vypne spínač - auto zastaví
            if (e.KeyCode == Keys.Right)
                timer1.Enabled = false;
        }
```

TYPOVÉ PŘÍKLADY:

1. Připravte podobnou aplikaci s obrázkem létajícího objektu, který se bude pohybovat vzhůru.