Středoškolská odborná činnost

Obor SOČ: 1. Matematika a statistika

Aplikace pro vykreslování grafů elementárních funkcí

Autor: Jan Procházka

Škola: Střední škola spojů a informatiky Tábor
Bydlinského 2474
390 11 Tábor

Kraj: Jihočeský kraj

Konzultant: Mgr. Jiřina Bartoňová

Tábor 2016

Prohlášení:

Prohlašuji, že jsem mojí práci SOČ vypracoval samostatně a použil jsem pouze podklady uvedené v seznamu vloženém v práci SOČ.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V Táboře dne Podpis:

Poděkování:

Děkuji Mgr. Jiřině Bartoňové za obětavou pomoc a podnětné připomínky, které mi během práce poskytovala.

ANOTACE

Cílem práce bylo vytvořit v jazyce C aplikaci, která vykresluje grafy elementárních funkcí, jak v základním tvaru, tak i s posunem ve směru os *x* a *y*.

Klíčová slova:

SOČ, matematika, grafy funkcí, elementární funkce, soustava souřadná, jednotka, C#, objektově orientované programování, Microsoft Visual Studio 2008

Obsah:

Úvod

Cílem práce bylo vytvořit aplikaci, která po zadání vstupních parametrů vykresluje grafy funkcí, jak v základním tvaru, tak i s posunem ve směru os *x* a *y*.

Při volbě tématu pro tuto práci jsem se inspiroval při hodinách matematiky ve druhém ročníku. Probírali jsme postupně různé funkce. U každé jsme sestrojovali její graf. Z grafu funkce v základním tvaru jsme pak odvozovali grafy funkcí, které vznikly přičtením konstanty k argumentu nebo hodnotě základní funkce. U některých funkcí jsme řešili i to, jak se změní graf funkce po vynásobení argumentu nebo hodnoty funkce.

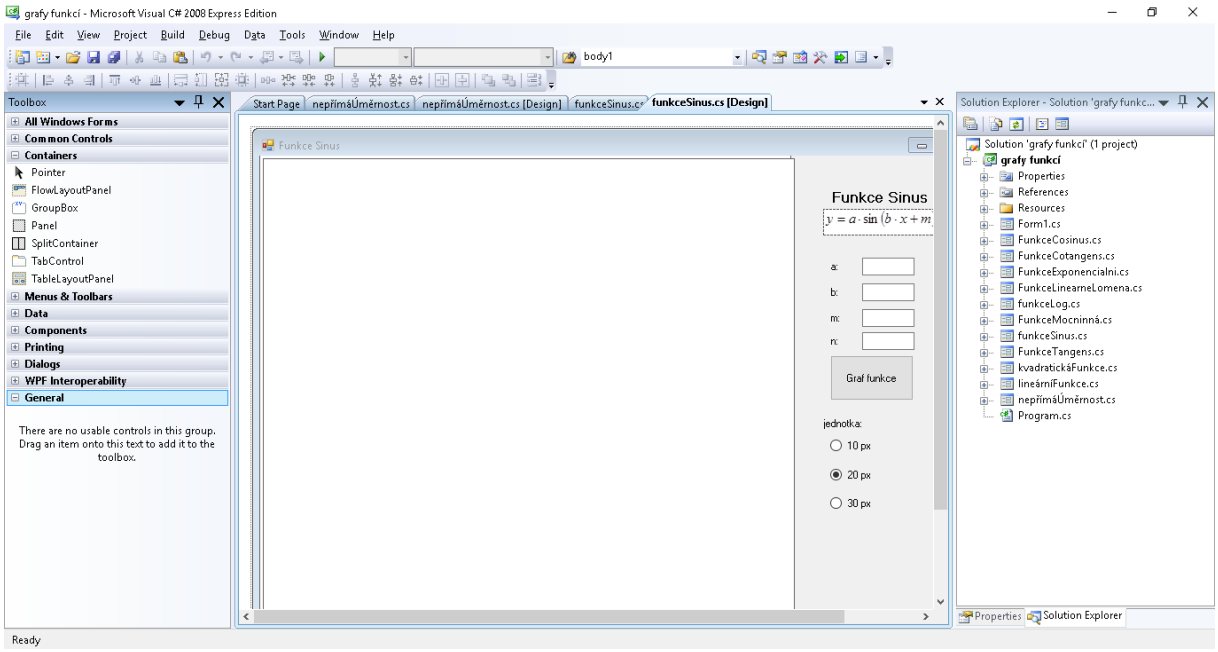
Pro vytvoření aplikace, která by se dala využít při výuce matematiky, jsem se rozhodl poté, co jsem se v hodinách programování seznámil s programovacím jazykem C#, hlavně s tvorbou grafiky v tomto jazyce.

1. Seznámení s programovacím jazykem a vývojovým prostředím

Microsoft Visual Studio je bohaté integrované vývojářské prostředí od společnosti Microsoft, ve kterém je možné vytvořit úžasné programy či aplikace a to například jazycích C#, C++, Basic, F#, Java, PHP a mnoho dalších.

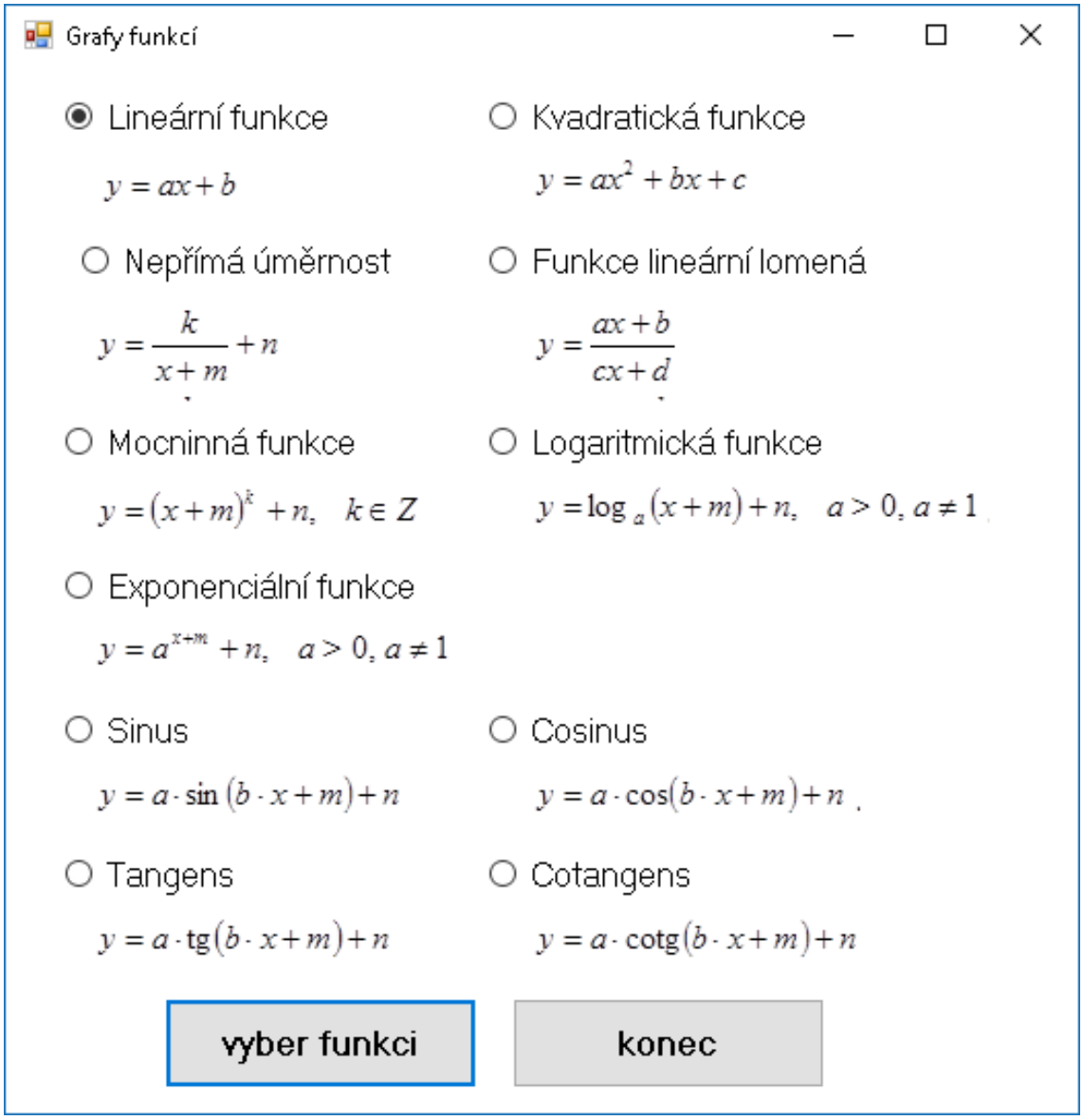
www.visualstudio.com

Aplikace je vytvořena v programovacím jazyce C#. Tento jazyk je určen pro tvorbu různých aplikací běžící v rozhraní NET Framework. Tento jazyk je jednoduchý, výkonný, přehledný a objektově orientovaný. Řada inovací v jazyce C# umožňuje rychlý vývoj aplikací a zároveň expresivity a elegance jazyků C.



Obr. 1 – vývojové prostředí

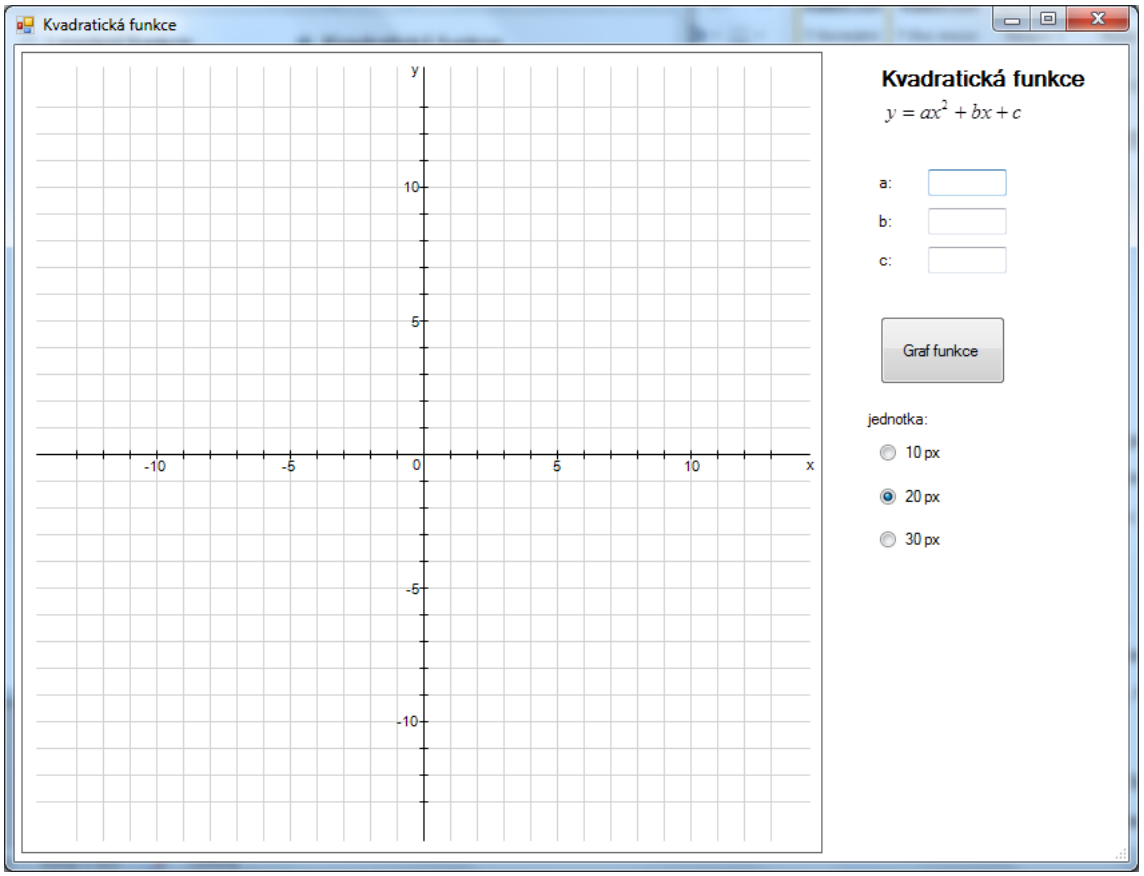
2. Úvodní formulář aplikace



Obr. 2 – úvodní formulář

3. Princip vykreslování grafu

3.1 Soustava souřadná a nastavení jednotky



Obr.3 –soustava souřadná v okně pro kvadratickou funkci

Pro vykreslení grafu funkce je použit ovládací prvek `Panel`. Velikost panelu je 600px na šířku i na výšku. K překreslení panelu dojde vždy, když nastane událost `Paint`. V obslužné metodě této události je vytvořena kreslící plocha s názvem `kp`.

```
Graphics kp=e.Graphics;
```

Tato plocha je totožná s celou plochou panelu. Při kreslení se používají souřadnice v pixelech. Levý horní roh má souřadnice [0;0]. Souřadnice ve vodorovném směru roste směrem doprava. Souřadnice ve svislém směru roste směrem dolů, tedy obráceně než v soustavě souřadné pro vykreslení grafu.

Do panelu se vykresluje soustava souřadná se všemi čtyřmi kvadranty. Středem této soustavy je přibližně střed panelu, tedy bod [300,300]. Jednotka je nastavena v celočíselné proměnné `jednotka`. Ve výchozím nastavení je hodnota této proměnné 20px. Pokud uživatel usoudí, že jednotku by bylo vhodné změnit, má možnost pomocí

přepínače v pravé části okna zvolit jednotku 10px nebo 30px a stiskem tlačítka „Graf funkce“ znovu vykreslit graf do nové soustavy souřadné. Nastavení jednotky je pomocí neúplných podmínek, ve kterých se testuje „zaškrtnutí“ jednotlivých přepínačů.

```
if (jednotkaDeset.Checked) jednotka = 10;
if (jednotkaDvacet.Checked) jednotka = 20;
if (jednotkaTřicet.Checked) jednotka = 30;
```

Soustava souřadná a následně i graf funkce se kreslí pomocí nástroje nazvaného `tužka`. Tato „tužka“ kreslí plnou čáru tloušťky 1px. V průběhu kreslení se mění pouze její barva.

```
Pen tužka = new Pen(Color.LightGray);
```

Nejprve se vykreslují světle šedé vodící čáry, poté černé souřadnicové osy, popisky na osách a čárky označující jednotky na jednotlivých osách. Vodící čáry a čárky na osách se vykreslují po čtyřech najednou, vždy v každém kvadrantu jedna. Počet vodících čar samozřejmě závisí na zvolené jednotce a prostoru, který máme pro každý kvadrant k dispozici.

```
int početČárek = 290 / jednotka;
```

Po vykreslení soustavy souřadné se testuje, zda byly zadány parametry pro funkci. Pokud ano, tak se tužkou červené barvy vykreslí křivka tvořená body zadanými v poli `body[]`.

```
if (zadáno)
{
   tužka.Color = Color.Red;
   kp.DrawCurve(tužka, body);
}
```

Pro „vyhlazení“ křivky je v úvodu metody příkaz:

```
kp.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;
```

Celý zdrojový kód metody `panelGraf_Paint` je součástí přílohy 1.

3.2 Graf funkce

Postup při vykreslování grafu budu vysvětlovat na příkladu okna pro kvadratickou funkci.

Graf funkce se vykreslí po zadání vstupních parametrů a stisknutí tlačítka „Graf funkce“. Pro toto tlačítko tímto nastane událost `Click`. V obslužné metodě pro tuto událost je nejprve nastavení jednotky, které jsem popsal v předchozí kapitole. Poté se z textových polí v pravé části okna načtou vstupní parametry. Zadané hodnoty jsou typu `string` (textový řetězec), ale pro výpočty je potřeba provést typovou konverzi na typ `double` (desetinné číslo). Konverze probíhají pomocí konstrukce `try – catch`. Ve větvi `try` se provede pokus o konverze a ve větvi `catch` se zachytí případná chyba a provádění metody se zastaví. K chybě dojde, pokud je v poli hodnota, kterou nelze převést na desetinné číslo, např. text nebo je textové pole prázdné.

```
try
{
        a = Convert.ToDouble(PoleA.Text);
        b = Convert.ToDouble(PoleB.Text);
        c = Convert.ToDouble(PoleC.Text);
}
catch
{
        MessageBox.Show("Chybně zadaná vstupní data", "CHYBA!");
        return;
}
```

Dále se, pokud je to potřeba, testují vstupní parametry. U kvadratické funkce musí být $a \neq 0$. Při správně zadaných vstupních parametrech se naplní pole body[]. Poté se logická proměnná `zadáno` nastaví na `true` a voláním metody `panelGraf.Refresh()` se vyvolá překreslení panelu s grafem funkce.

Pro vykreslování křivky grafu funkce metodou `DrawCurve()` je potřeba vytvořit pole bodů. Pokud je graf tvořen více křivkami (např. dvě větve hyperboly u nepřímé úměrnosti nebo jednotlivé křivky u tangens a kotangens), pak musíme vytvořit pole pro každou křivku. Pole je datová struktura s daným počtem složek stejného typu, které se vzájemně rozlišují indexem. V jazyce C# se indexuje celými čísly od 0.

Souřadnice bodů v poli jsou v pixelech vzhledem k souřadnicím panelu. Při výpočtu souřadnic nejprve musíme vypočítat bod v souřadnicích *x*, *y* vzhledem

k soustavě souřadné pro vykreslení grafu. K tomu slouží reálné proměnné `x` a `y`. Z nich poté vytvoříme hodnoty proměnných `xpx` a `ypx`.

```
xpx = (int) (x * jednotka + 300);
ypx = (int) (-y * jednotka + 300);
```

Dále vytvoříme bod, který zapíšeme do pole `body[]` na pozici danou proměnnou `index`.

```
Point bod = new Point(xpx,ypx);
body[index]=bod;
```

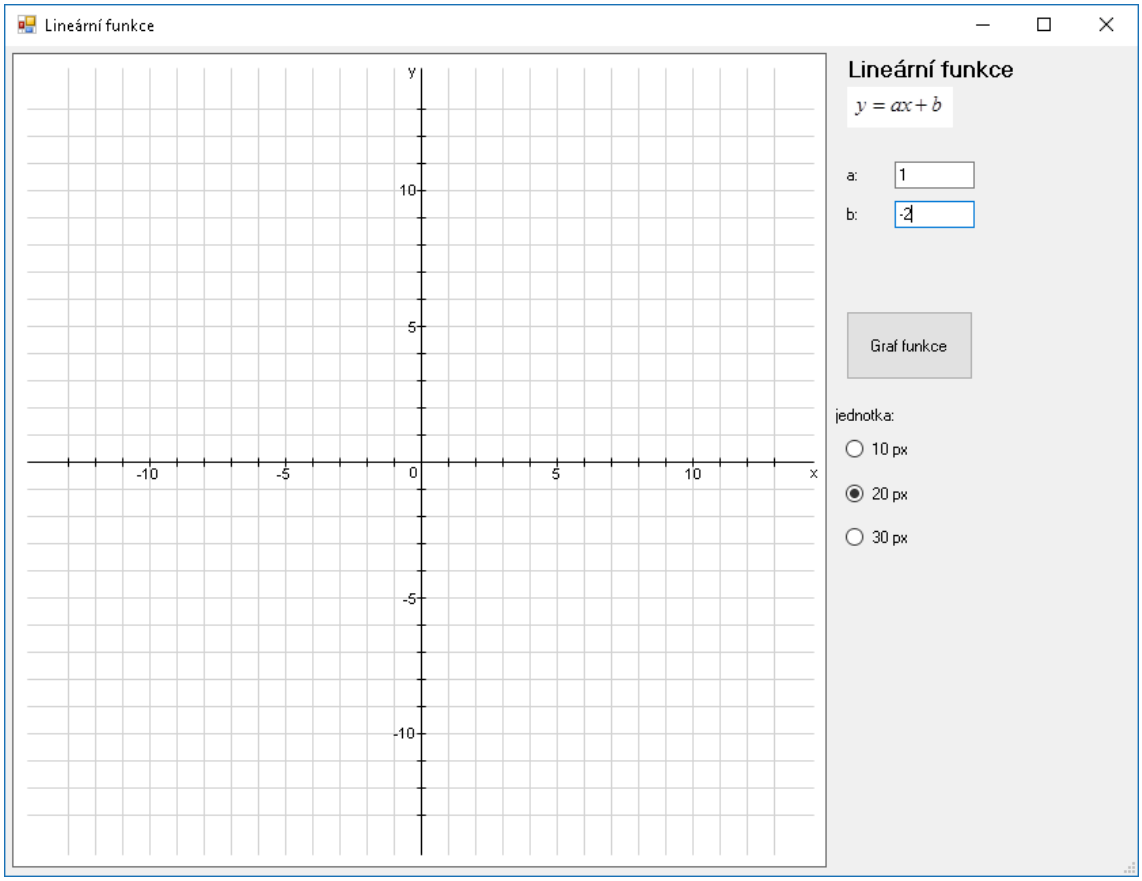
Hodnotu proměnné x nastavíme na začátek intervalu, ve kterém vykreslujeme graf. Konkrétně u kvadratické funkce je to -4 jednotky od vrcholu paraboly. Po dopočítání `y` podle předpisu funkce a vytvoření bodu v pixelech se `x` zvětší o 0,5 jednotky. Cyklus pro výpočet `x` končí +4 jednotky od vrcholu. Vytvoříme tak pole o celkem 17 bodech (8 doleva od vrcholu, vrchol a 8 doprava). Pro každou funkci je vytvořeno pole o jiném počtu bodů, protože se liší intervaly, ve kterých se funkce kreslí a také velikost posunu `x`.

Celý zdrojový kód metody `tlačítkoKresliGraf_Click` je součástí přílohy.

4. Okna jednotlivých funkcí

4.1 Lineární funkce

Předpis lineární funkce je $y = ax + b$. Grafem lineární funkce je přímka. Graf se vykresluje jako úsečka s body pro $x_1 = -\frac{b}{a} - 10$ a $x_2 = x_1 + 20$. Tedy -± 10 jednotek od průsečíku s osu x.

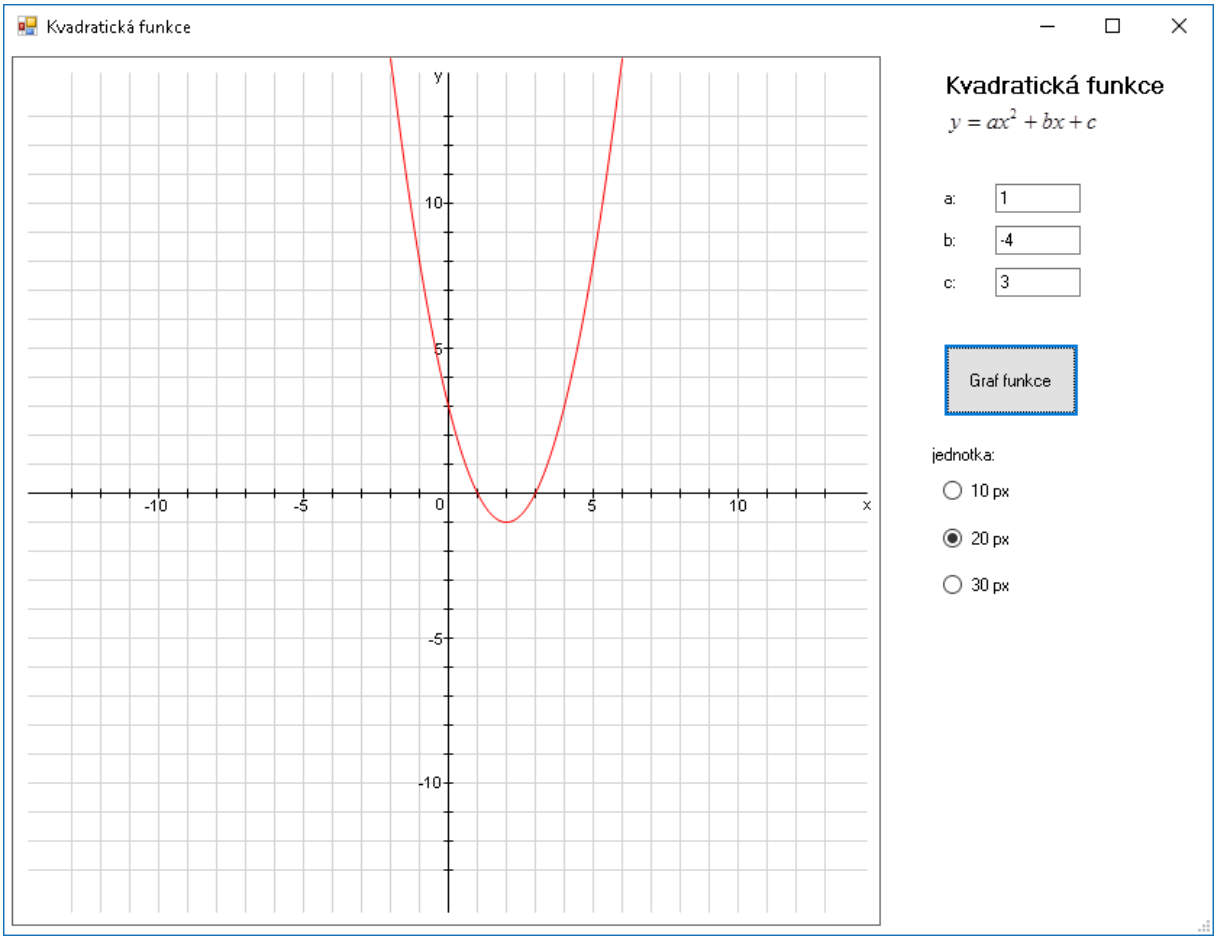


Obr.4 –graf lineární funkce

4.2 Kvadratická funkce

Předpis kvadratické funkce je $y = ax^2 + bx + c$, kde $a \neq 0$. Tuto skutečnost jsem zohlednil v testování podmínky pro zadané a

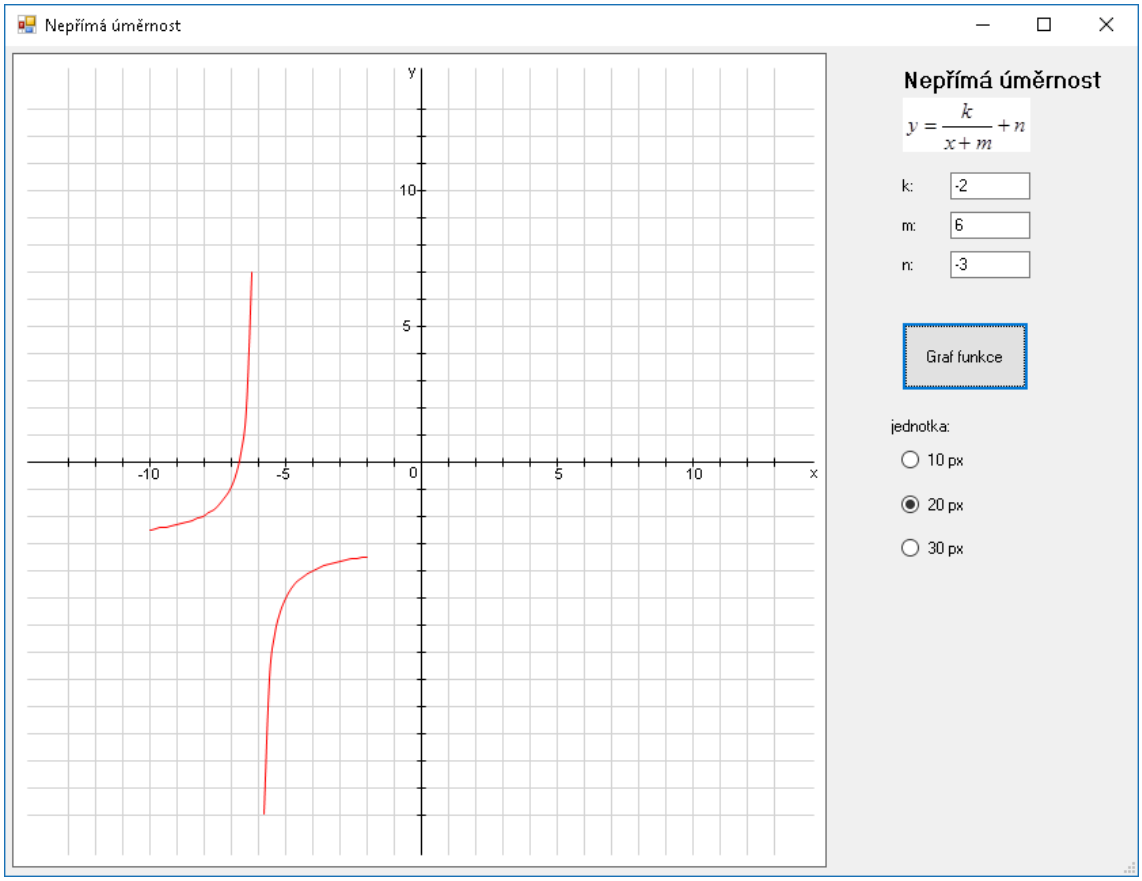
Grafem kvadratické funkce je parabola. Pole pro vykreslení křivky má 17 bodů. Počáteční x je nastaveno na -4 jednotky od vrcholu paraboly $-\frac{b}{2a}$. Posun je o 0,5 jednotky.



Obr. č. 5-graf kvadratické funkce

4.3 Nepřímá úměrnost

Předpis nepřímé úměrnosti je $y = \frac{k}{x+m} + n$ Grafem nepřímé úměrnosti je rovnoosá hyperbola. Nepřímá úměrnost je na rozdíl od předchozích funkcí vykreslována ne jednou, ale dvěma křivkami. Počítání pro x je nastaveno -4 jednotky od vrcholu hyperboly. Posun je 0,2 jednotky.

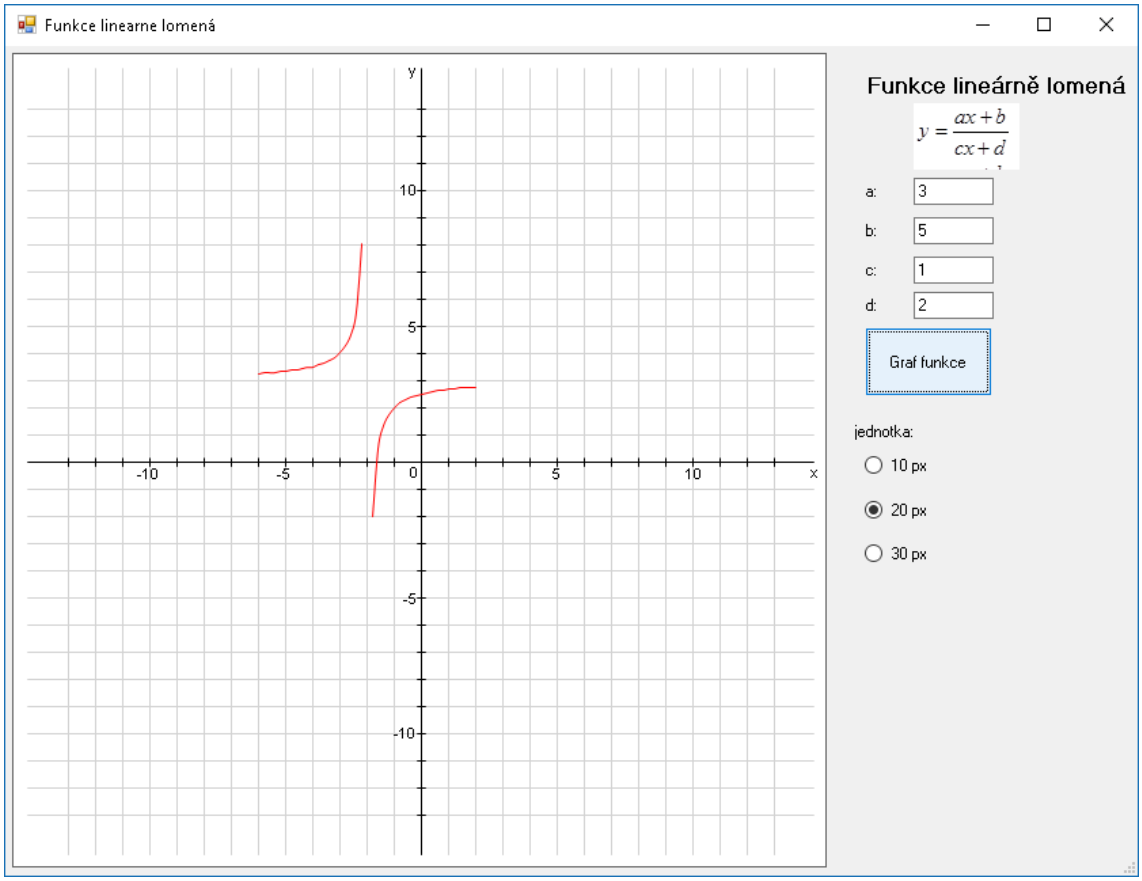


Obr. č. 6 - graf nepřímé úměrnosti

4.4 Funkce lineární lomená

Předpis funkce je $x = \frac{ax+b}{cx+d}$, kde $c \neq 0$. Tuto skutečnost jsem zohlednil v testování podmínky pro zadané c.

Grafem funkce je hyperbola. Funkce lineárně lomená se stejně jako nepřímá úměrnost vykresluje dvěma křivkami. Pole pro vykreslení křivky mají 20 bodů. Pro levou křivku je x nastaveno $x = \frac{-d}{c} - 4$. Posun je o 0,2 jednotky. Pravá křivka začíná 0,2 jednotky od nulového bodu, tedy $x = \frac{-d}{c} + 0{,}2$



Obr. č. 7 - graf lineárně lomené funkce

4.5 Mocninná funkce

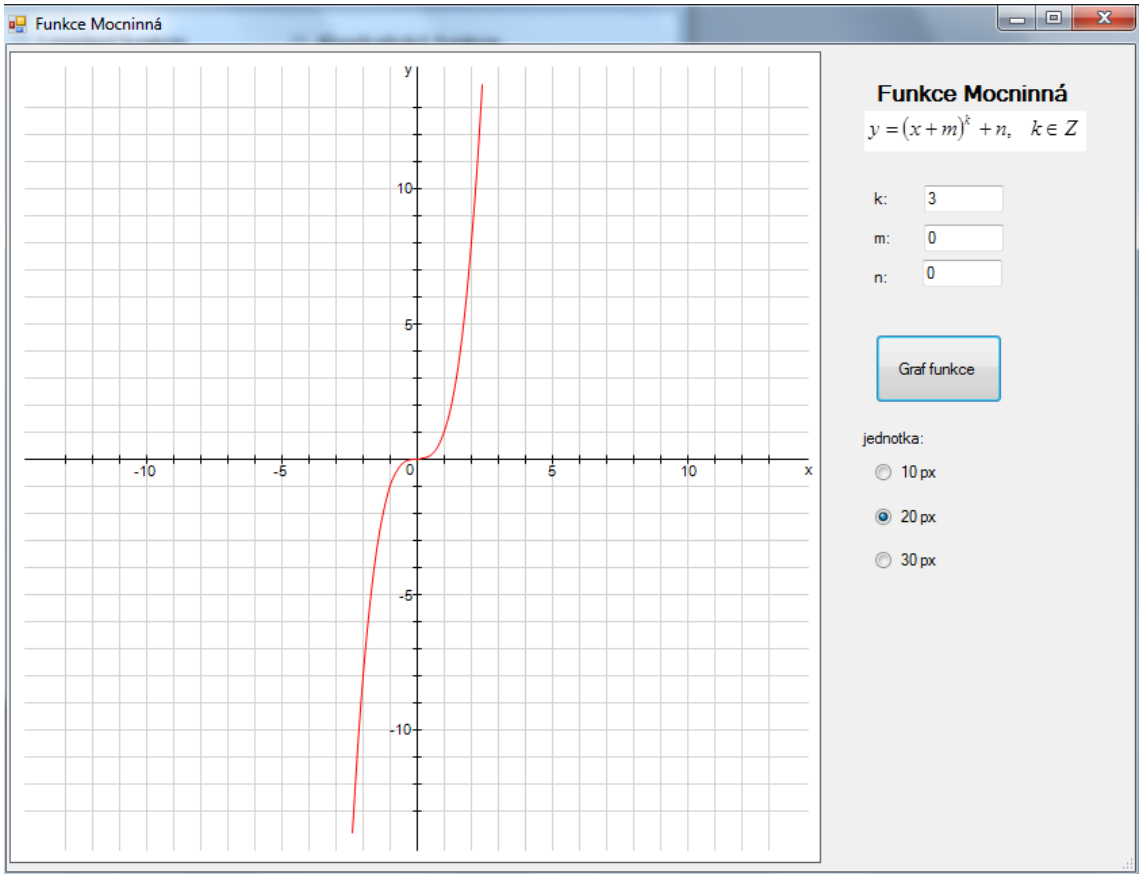
Předpis mocninné funkce je $y = (x + m)^k + n$, kde $k \epsilon Z, k \neq 0$. Tuto skutečnost jsem zohlednil v testování podmínky pro zadané *k* a datovým typem proměnné *k*. Pro k>0 je grafem funkce jedna křivka, při jejím vykreslení postupujeme stejně jako u kvadratické funkce. Pro k<0 tvoří graf funkce dvě křivky, při jejich vykreslení postupujeme stejně jako u nepřímé úměrnosti.

Při testování mocninné funkce se vyskytl problém. Pro k>3 měla křivka nesprávný tvar. Chyba byla způsobena velmi vysokými nebo velmi malými funkčními hodnotami, které způsobily „přetečení“ proměnné a k tím chybnému výpočtu souřadnice bodu křivky. Pro vyřešení tohoto problému jsem pro tuto funkci musel zvolit jinou datovou strukturu pro zadání bodů. Statickou datovou strukturu pole body [] jsem nahradil dynamickou datovou strukturou `seznamBodů`.

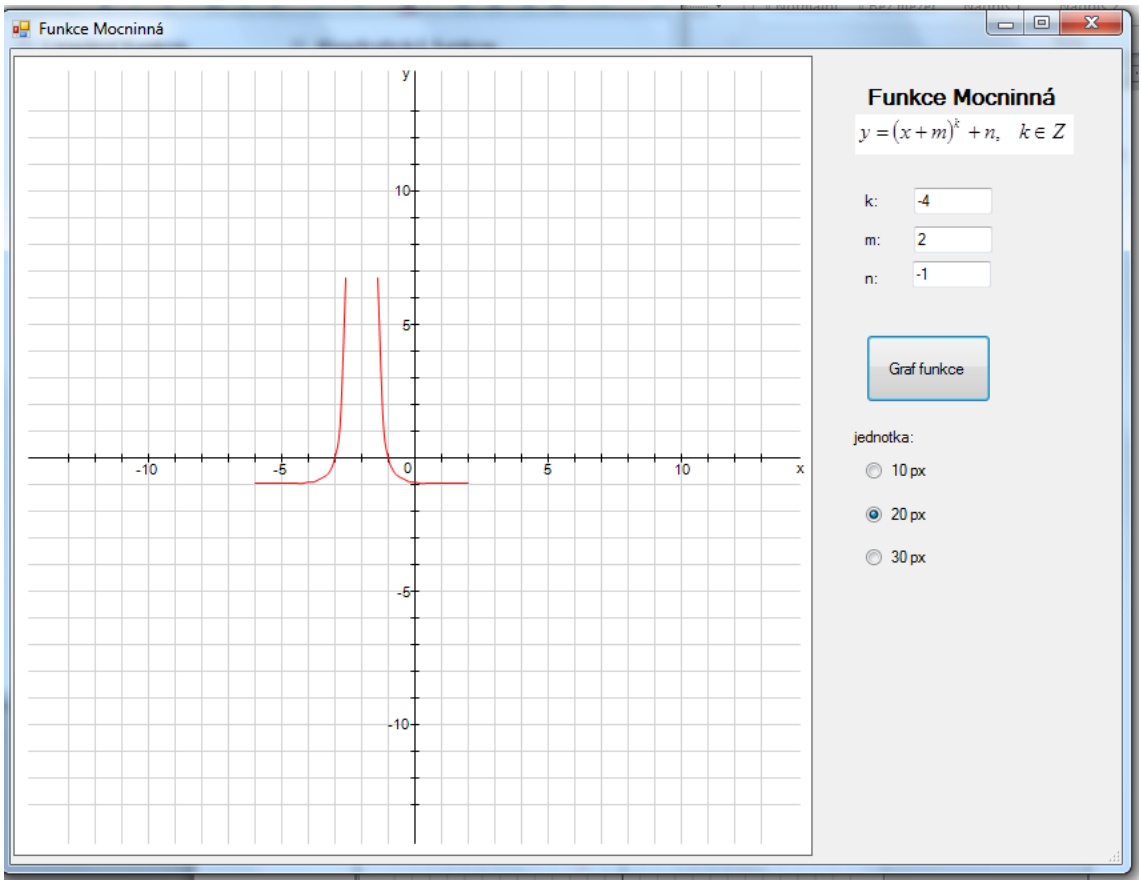
```
List<Point> seznamBodů=new List<Point>();
```

Pole bylo nevyhovující, protože má pevnou délku, kterou je nutné zadat již při deklaraci. Datová struktura seznam je dynamická, to znamená, že nemusíme předem vědět, kolik bodů bude obsahovat. Můžeme do něj body vkládat, až po otestování, zda se nám vejdou do vykreslované části grafu. Metoda pro vykreslení křivky bohužel umí pracovat pouze s polem. Je proto nutné před vykreslením zjistit délku seznamu, deklarovat stejně velké pole a seznam převést na pole.

```
int početBodů=seznamBodů.Count();
Point[] body = new Point[početBodů];
for (int index = 0; index < početBodů; index++)
body[index] = seznamBodů[index];
kp.DrawCurve(tužka, body);
seznamBodů=new List<Point>();
```



Obr. č. 8 - graf mocninné funkce pro k>0

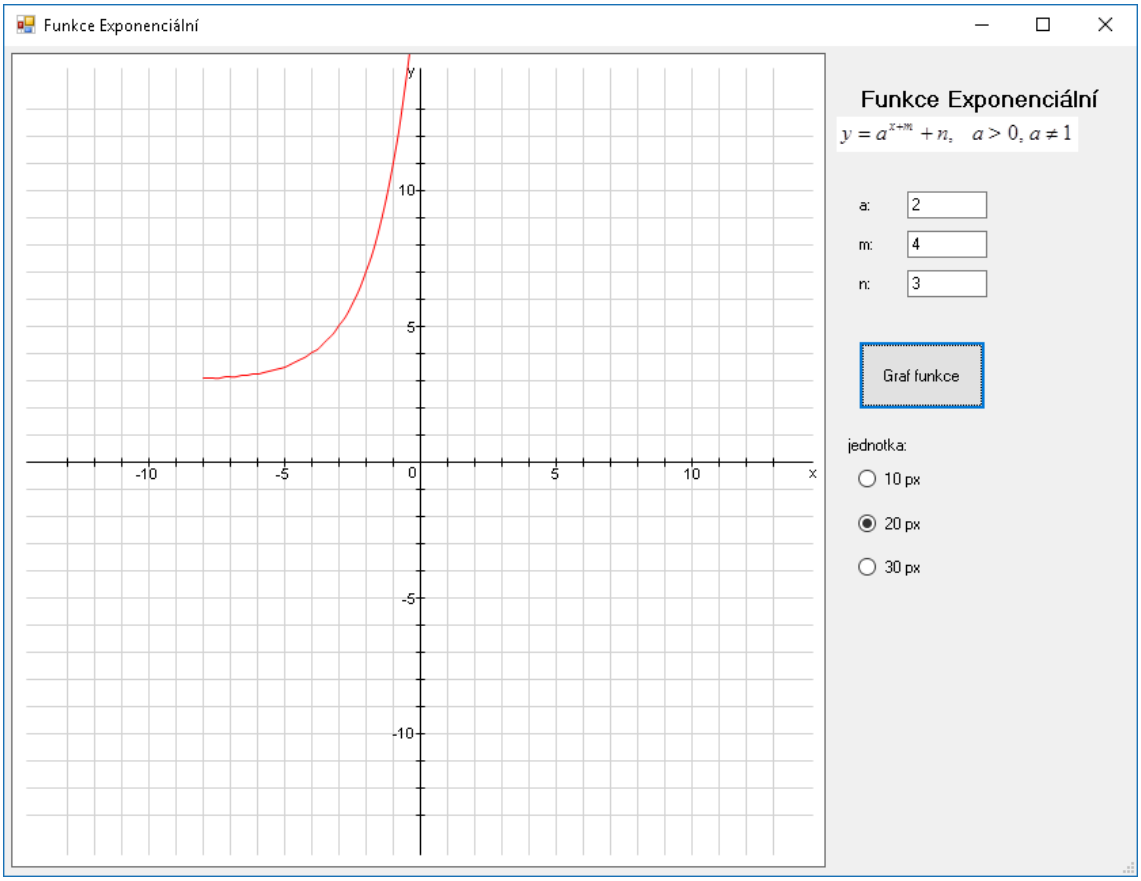


Obr. č.9 - graf mocninné funkce pro k<0

4.6 Exponenciální funkce

Předpis exponenciální funkce je $y = a^{x+m} + n$, kde $a > 0, a \neq 1$. Tuto skutečnost jsem zohlednil v testování podmínky pro zadané *a*.

Grafem exponenciální funkce je exponenciální křivka. Pole pro vykreslení křivky má 41 bodů. Počítání pro x je nastaveno na -4 od jednotky od bodu –*m* exponenciální křivky. Posun je o 0,2 jednotky.

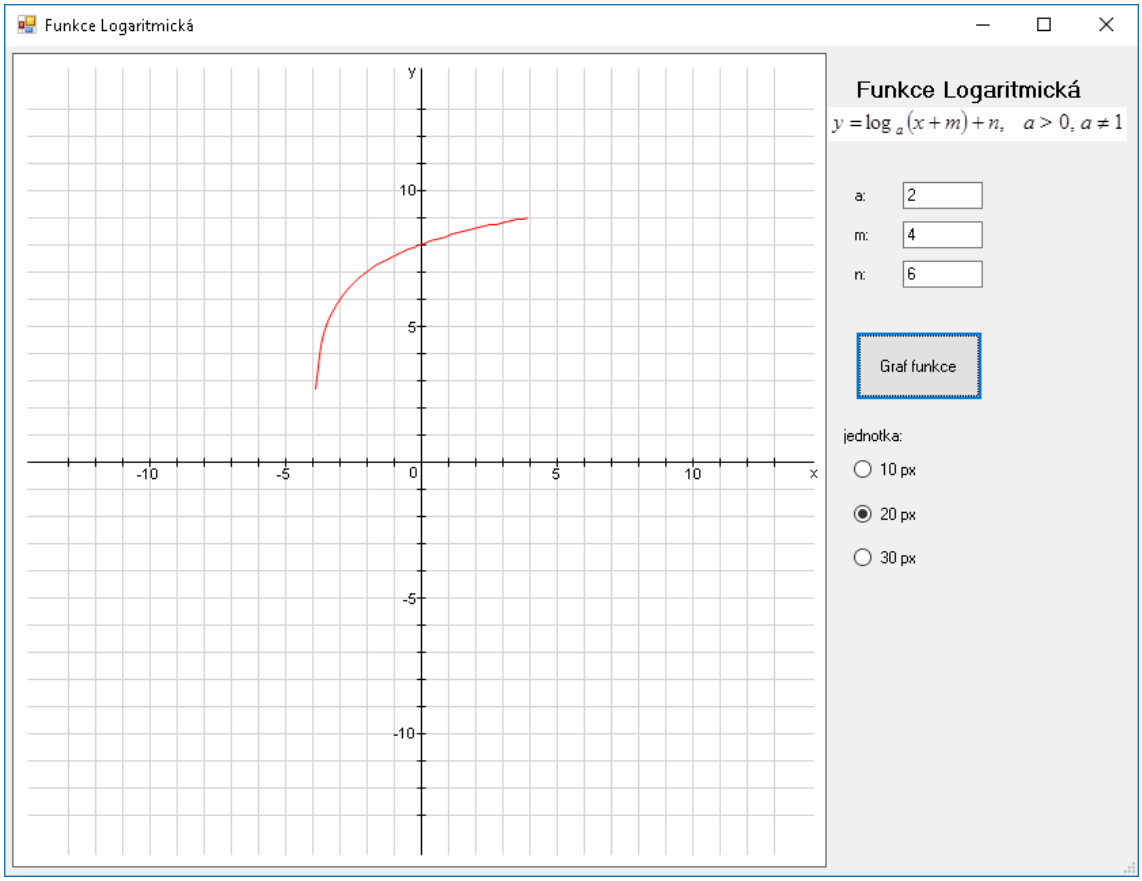


Obr. č. 10 - graf exponenciální funkce

4.7 Logaritmická funkce

Předpis logaritmické funkce je $y = log_a\ (x + m) + n$, kde $a > 0, a \neq 1$. Tuto skutečnost jsem zohlednil v testování podmínky pro zadané a.

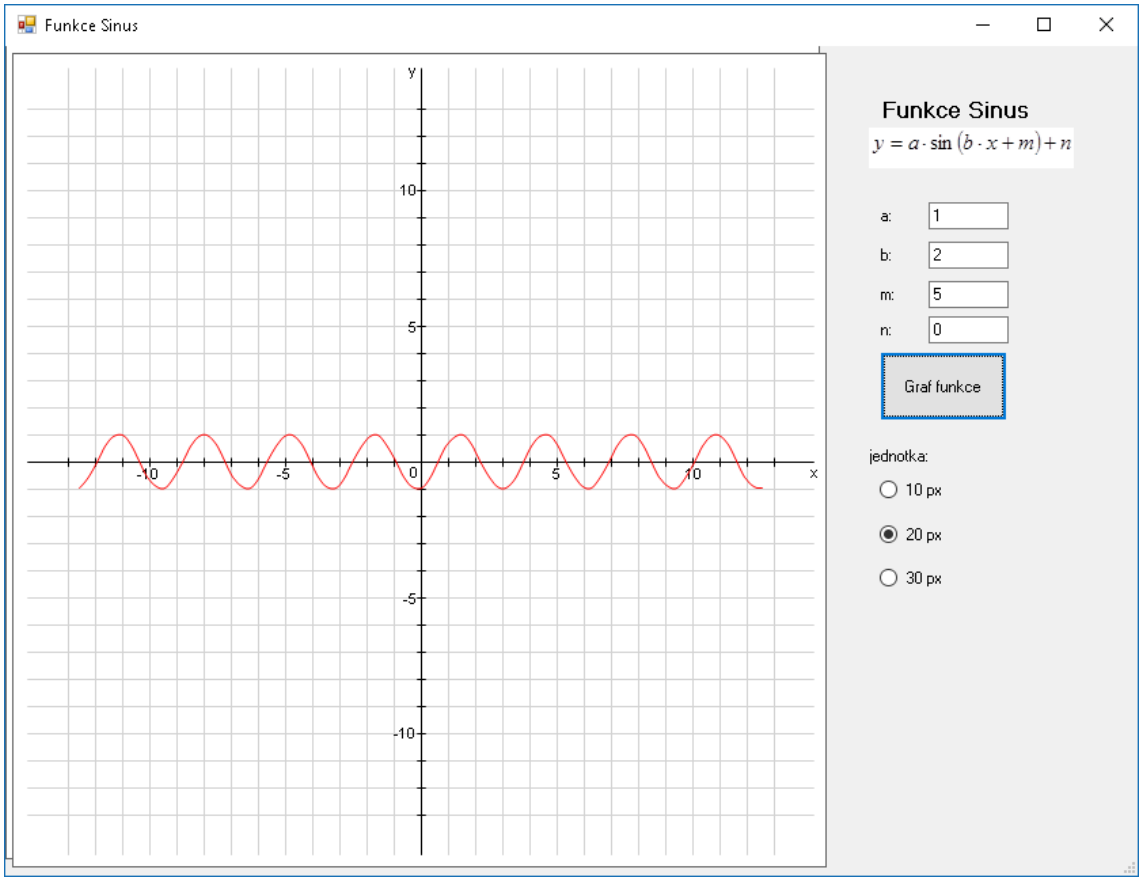
Grafem logaritmické funkce je logaritmická křivka. Pole pro vykreslení obsahuje 40 bodů. Počáteční x je nastaveno na 0,1 jednotky od $-m$. Posun je o 0,2 jednotky.



Obr. č. 11 - graf logaritmické funkce

4.8 Sinus

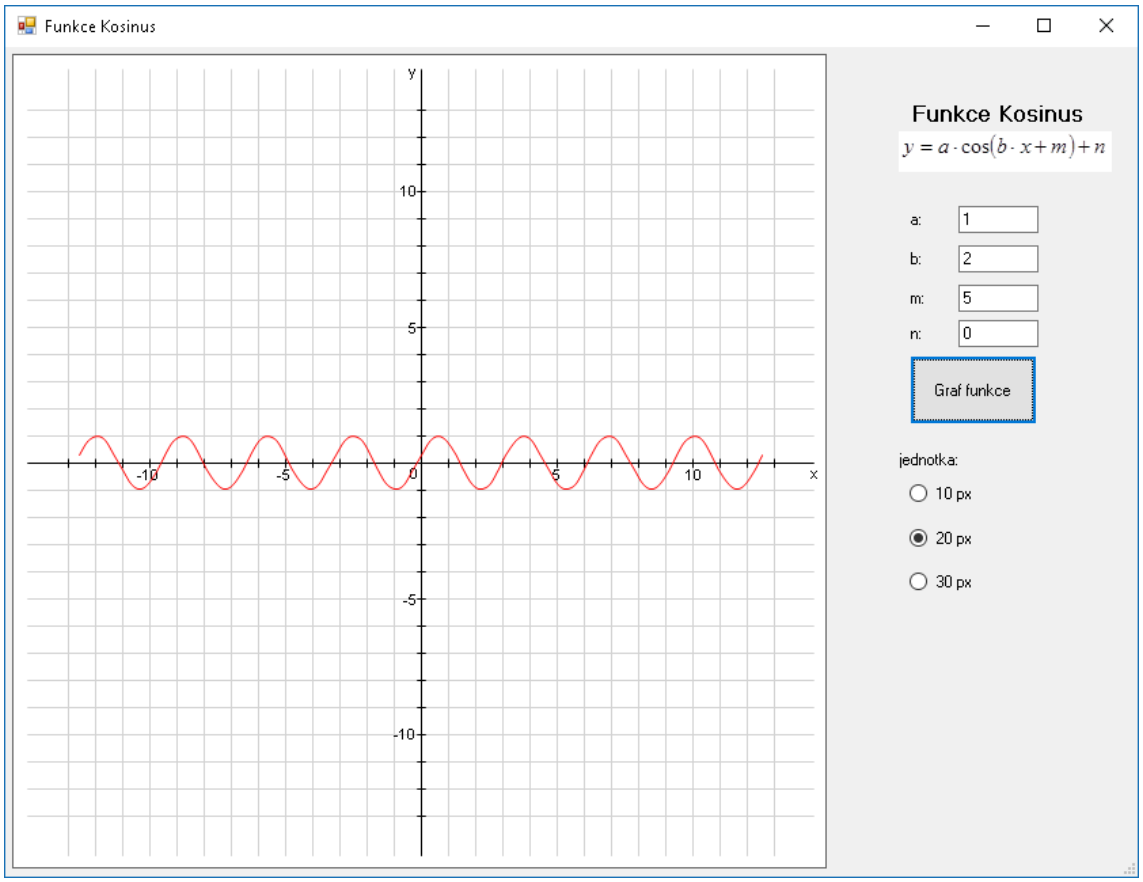
Předpis funkce sinus je $y = a * \sin(b * x + m) + n$. Grafem funkce sinus je sinusoida. Pole pro vykreslení křivky má 81 bodů. Počáteční x je nastaveno na -4 π od vrcholu sinusoidy. Posun je o 0,1 π jednotky.



Obr. č. 12 - graf funkce sinus

4.9 Kosinus

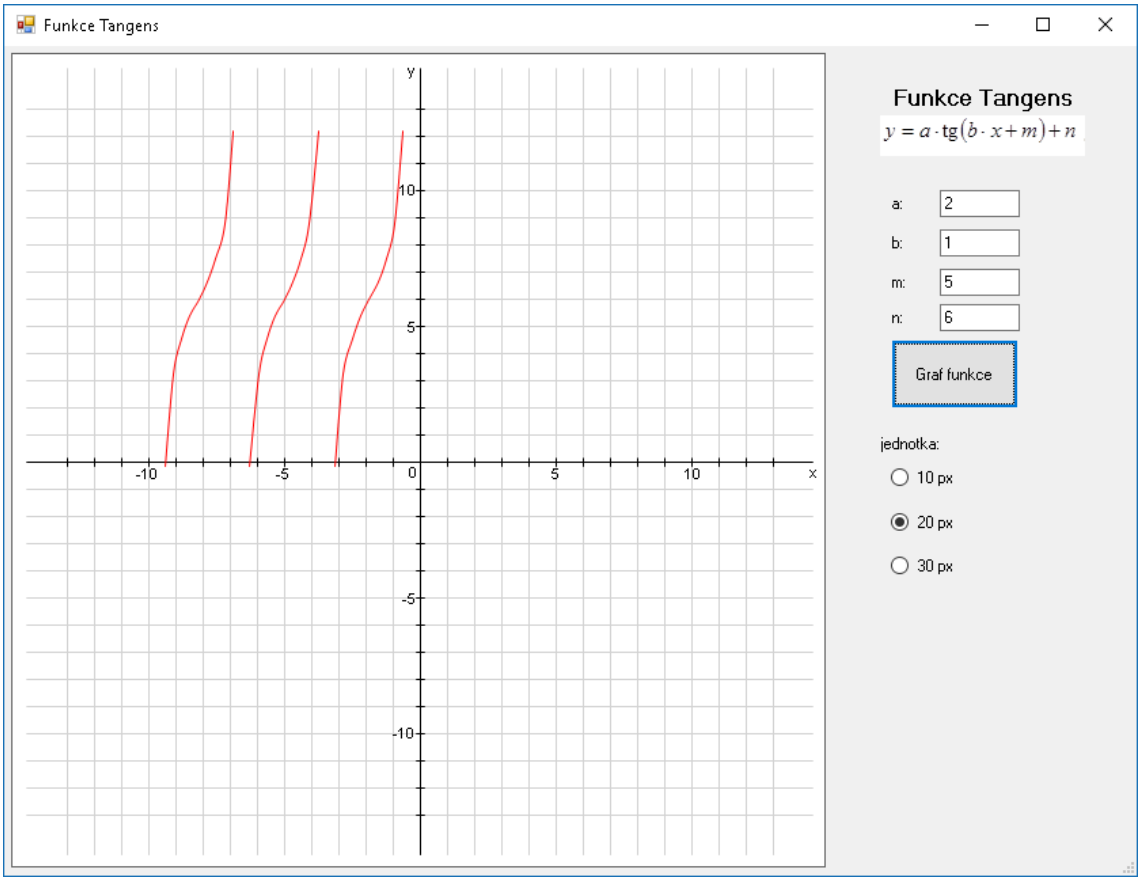
Předpis funkce kosinus je $y = a * \cos(b * x + m) + n$. Grafem funkce kosinus je kosinusoida. Pole pro vykreslení křivky má 81 bodů. Počáteční x je nastaveno na -4 π od vrcholu sinusoidy. Posun je o 0,1 π jednotky.



Obr. č. 13 - graf funkce kosinus

4.10 Tangens

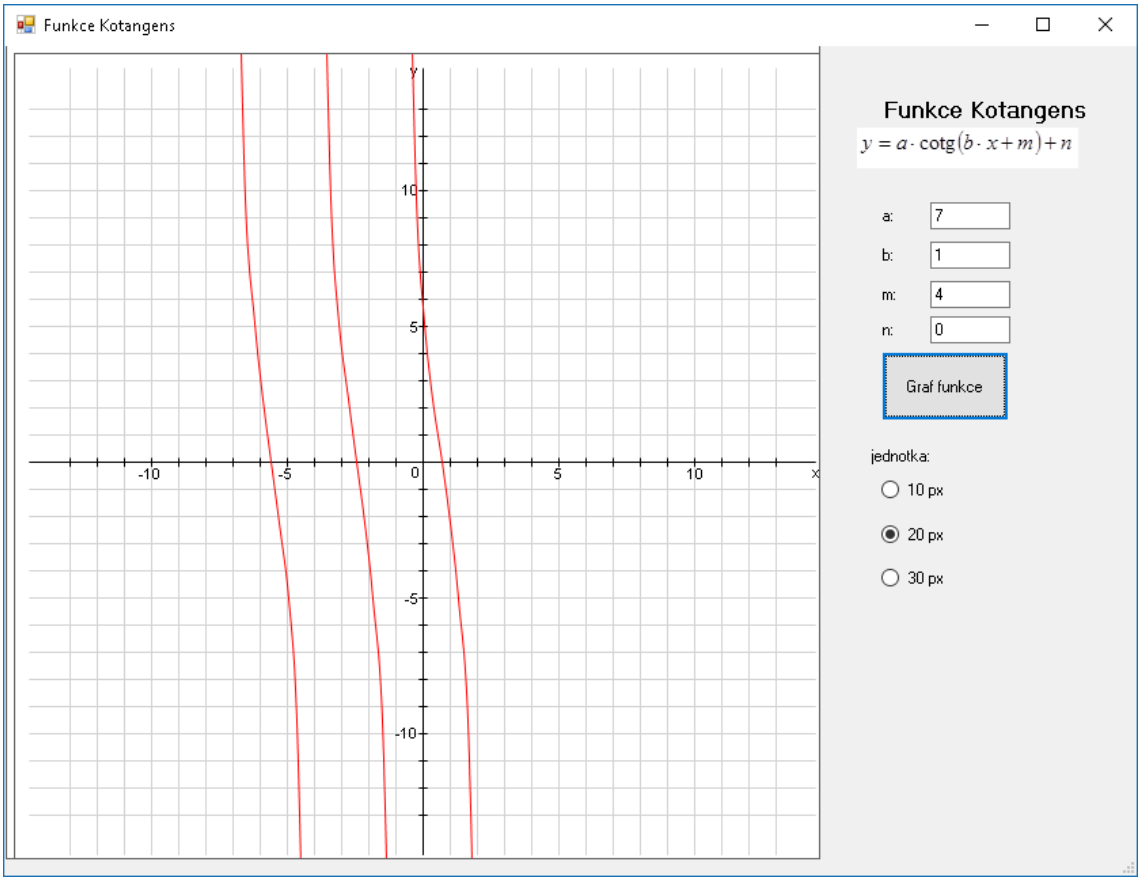
Předpis funkce je $y = a * tg(b * x + m) + n$. Grafem funkce je nekonečně mnoho křivek posunutých o celočíselné násobky π. Pole pro vykreslení křivky v intervalu $(-\frac{\pi}{2} - m; \frac{\pi}{2} - m)$ má 9 bodů. Počáteční x je nastaveno na -0,4π jednotky od bodu –m. Posun je o 0,1 π jednotky. Celkem se vykreslují 3 křivky. Pole bodů pro posunuté křivky se vytvářejí z pole pro původní křivku tím, že se x-ová souřadnice všech bodů zvětší nebo zmenší o π.



Obr. č. 14 - graf funkce tangens

4.11 Kotangens

Předpis funkce je $y = a * cotg(b * x + m) + n$. Grafem funkce je nekonečně mnoho křivek posunutých o celočíselné násobky π. Pole pro vykreslení křivky v intervalu $(-m; \pi - m)$ má 9 bodů. Počáteční x je nastaveno na +0,1π jednotky od bodu –m. Posun je o 0,1 π jednotky. Celkem se vykreslují 3 křivky. Pole bodů pro posunuté křivky se vytvářejí z pole pro původní křivku tím, že se x-ová souřadnice všech bodů zvětší nebo zmenší o π. Hodnoty funkce kotangens se počítají jako převrácená hodnota funkce tangens. Důvodem je chybějící funkce pro kotangens v třídě matematických funkce `Math()` .



Obr. č. 15 - graf funkce kotangens

Závěr

Při tvorbě aplikace jsem si osvěžil znalosti o grafech funkcí. Zároveň jsem si poprvé vyzkoušel vytvořit kompletní aplikaci od návrhu vzhledu a návrhu algoritmu až po realizaci a testování funkčnosti aplikace na vzorových datech.

Do aplikace jsem zahrnul všechny funkce, o kterých jsme se učili. Zároveň jsem chtěl zpracovat i všechny možné změny grafu funkce, tedy co se s grafem funkce děje při přičítání a násobení argumentu nebo hodnoty funkce. Jediné co jsem v aplikaci neřešil, je vliv absolutní hodnoty na graf funkce. K určení absolutní hodnoty slouží funkce `Math.Abs()`. Její použití není problém. Nevím si ovšem rady se způsobem zadání vstupních dat, konkrétně s rozsahem platnosti absolutní hodnoty.

Aplikaci jsem předvedl vyučujícím matematiky na naší škole. Hodnotili ji jako zajímavou a chtějí tuto aplikaci využívat při výuce.

Přílohy

Příloha 1

Zdrojový kód pro okno „Kvadratická funkce“

```
public partial class kvadratickáFunkce : Form
{
   double a = 0;
   double b = 0;
   double c = 0;
   bool zadáno = false;
   Point [] body = new Point [17];
   int jednotka = 20;

   public kvadratickáFunkce()
   {
      InitializeComponent();
   }

    private void tlačítkoKresliGraf_Click(object sender, EventArgs e)
    {
       if (jednotkaDeset.Checked) jednotka = 10;
       if (jednotkaDvacet.Checked) jednotka = 20;
       if (jednotkaTřicet.Checked) jednotka = 30;
       try
       {
           a = Convert.ToDouble(PoleA.Text);
           b = Convert.ToDouble(PoleB.Text);
           c = Convert.ToDouble(PoleC.Text);
       }
       catch
       {
           MessageBox.Show("Chybně zadaná vstupní data", "CHYBA!");
           return;
       }
       if (a == 0)
           MessageBox.Show("Chybně zadaná vstupní data", "CHYBA!");
       else
       {
           double vrcholX = -b / (2 * a);
           double x = vrcholX - 4;
           double y;
           int xpx, ypx;
           for (int index = 0; index <= 16; index++)
           {
               y = a * x * x + b * x + c;
               xpx = (int) (x * jednotka + 300);
               ypx = (int) (-y * jednotka + 300);
               Point bod = new Point(xpx,ypx);
               body[index]=bod;
               x += 0.5;
           }
           zadáno = true;
           panelGraf.Refresh();
        }
   }
```

```
private void panelGraf_Paint(object sender, PaintEventArgs e)
{
   Graphics kp=e.Graphics;
  kp.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;
   Pen tužka = new Pen(Color.LightGray);
   int početČárek = 290 / jednotka;
   for (int index = 1; index < početČárek; index++)
   {
      kp.DrawLine(tužka,300+index*jednotka,10,300+index*jednotka,589);
      kp.DrawLine(tužka,300-index*jednotka,10,300-index*jednotka,589);
      kp.DrawLine(tužka,10,300+index*jednotka,589,300+index*jednotka);
      kp.DrawLine(tužka,10,300-index*jednotka,589,300-index*jednotka);
   }
   tužka.Color = Color.Black;
   kp.DrawLine(tužka, 10, 300, 589, 300);
   kp.DrawLine(tužka, 300, 10, 300, 589);
   kp.DrawString("0", new Font("Arial", 8), Brushes.Black, 290, 301);
   kp.DrawString("-10", new Font("Arial", 8), Brushes.Black,
   69 + jednotka, 302); //číslo -10 na X
   kp.DrawString("-5", new Font("Arial", 8), Brushes.Black,
   172 + jednotka, 302); //číslo -5 na X
   kp.DrawString("5", new Font("Arial", 8), Brushes.Black,
   375 + jednotka, 302); //číslo 5 na X
   kp.DrawString("10", new Font("Arial", 8), Brushes.Black,
   473 + jednotka, 302); //číslo 10 na X
   kp.DrawString("-10", new Font("Arial", 8), Brushes.Black, 278,
   513 - jednotka); //číslo -10 na Y
   kp.DrawString("-5", new Font("Arial", 8), Brushes.Black, 285,
   414 - jednotka); //číslo -5 na Y
   kp.DrawString("5", new Font("Arial", 8), Brushes.Black, 289,
   214 - jednotka); //číslo 5 na Y
   kp.DrawString("10", new Font("Arial", 8), Brushes.Black, 283,
   113 - jednotka); //číslo 10 na Y
   kp.DrawString("x", new Font("Arial", 8), Brushes.Black, 585, 301);
   kp.DrawString("y", new Font("Arial", 8), Brushes.Black, 289, 5);

    for (int index = 1; index < početČárek; index++)
    {
       kp.DrawLine(tužka, 300 + index * jednotka, 297,
       300 + index * jednotka, 303);
       kp.DrawLine(tužka, 300 - index * jednotka, 297,
       300 - index * jednotka, 303);
       kp.DrawLine(tužka, 297,300 + index * jednotka, 303,
       300 + index * jednotka);
       kp.DrawLine(tužka, 297, 300 - index * jednotka, 303,
       300 - index * jednotka);
    }

    if (zadáno)
    {
       tužka.Color = Color.Red;
       kp.DrawCurve(tužka, body);
    }
  }
}
```

Příloha 2

Obsah vloženého CD

- Text práce ve formátu PDF.
- Složka s kompletním zdrojovým kódem aplikace.
- Samostatně spustitelná aplikace

Zdroje

www.matematika.cz

www.visualstudio.com

www.msdn.microsoft.com

ODVÁRKO, Oldřich. Matematika pro gymnázia, Funkce. Prometheus,2007.
ISBN 978-80-7196-164-2

ODVÁRKO, Oldřich. Matematika pro gymnázia, Goniometrie. Prometheus, 1994.
ISBN 80-85-849-57-7