Středoškolská odborná činnost

Obor SOČ: 1. Matematika a statistika

GRAFY FUNKCÍ

Autor: Jan Procházka

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

Kraj: Jihočeský kraj

Konzultant: Mgr. Jiřina Bartoňová

Tábor 2017

Prohlášení

Prohlašuji, že jsem svou 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 12.4.2017 Jan Procházka

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 pro vykreslování grafů funkcí, jak v základním tvaru, tak s posunem ve směru os x a y. Aplikace kromě vykreslování grafů nabízí i informace o vlastnostech každé ze zmiňovaných funkcí.

Klíčová slova

Grafy funkcí, soustava souřadná, jednotka, C#, objektově orientované programování, debugging

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*. Vycházel jsem ze své práce „Aplikace pro vykreslování grafů elementárních funkcí“ z loňského ročníku.

Tehdy vytvořenou aplikaci jsem použil jako předlohu pro novou aplikaci napsanou v nové verzi vývojového prostředí Visual Studio 2015. Výrazně jsem rozšířil funkce úvodního formuláře, především o poskytování informací o zvolené funkci. Při vykreslování grafů jednotlivých funkcí jsem napravil chyby, které vznikaly při změně jednotky zobrazení. Grafy jsem vykresloval v celém rozsahu viditelné části definičního oboru. U všech funkcí jsem vytvářel seznam bodů a teprve před vykreslením křivky jsem z něj vytvářel pole. Dále jsem odstranil běhovou chybu způsobenou vytvářením pole bodů z prázdného seznamu bodů. K této chybě došlo, pokud byl celý graf funkce mimo vykreslovací oblast nebo minimalizaci okna případně při stisku klávesy ALT.

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.

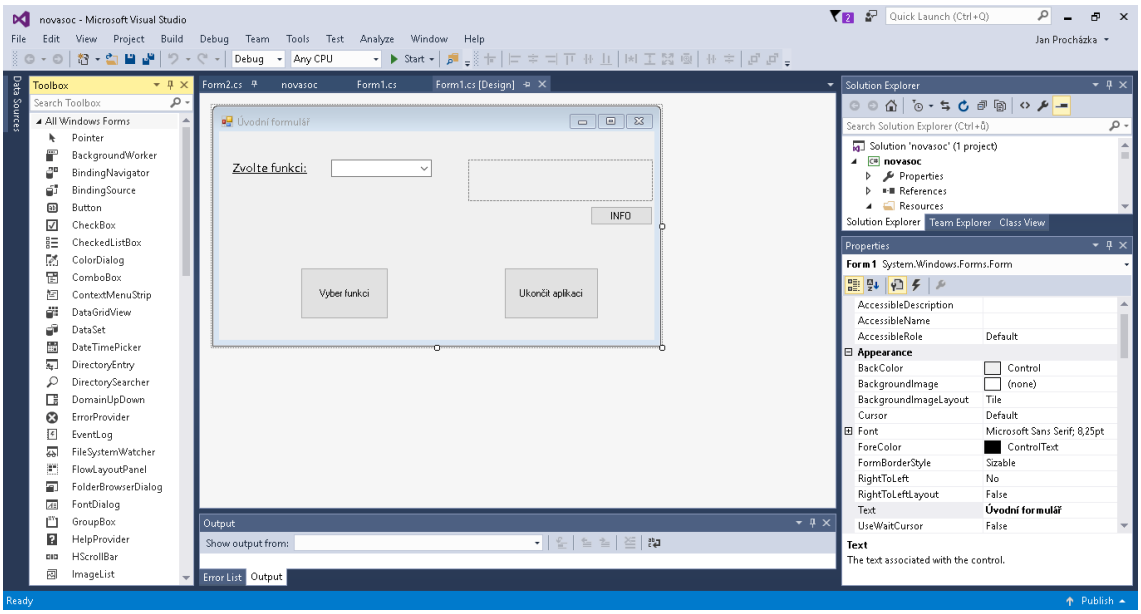
Pro vytvoření aplikace, která by se dala využít při výuce matematiky, jsem se rozhodl poté, co jsem se v hodinách programování seznámil s programovacím jazykem C#, hlavně s tvorbou grafiky v tomto jazyce.

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.

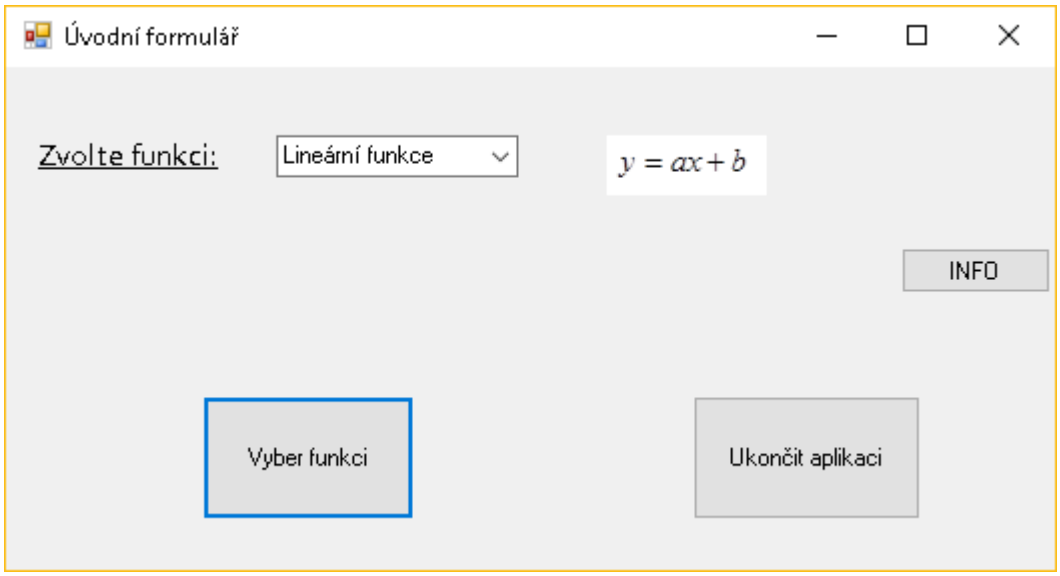
www.visualstudio.com

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.



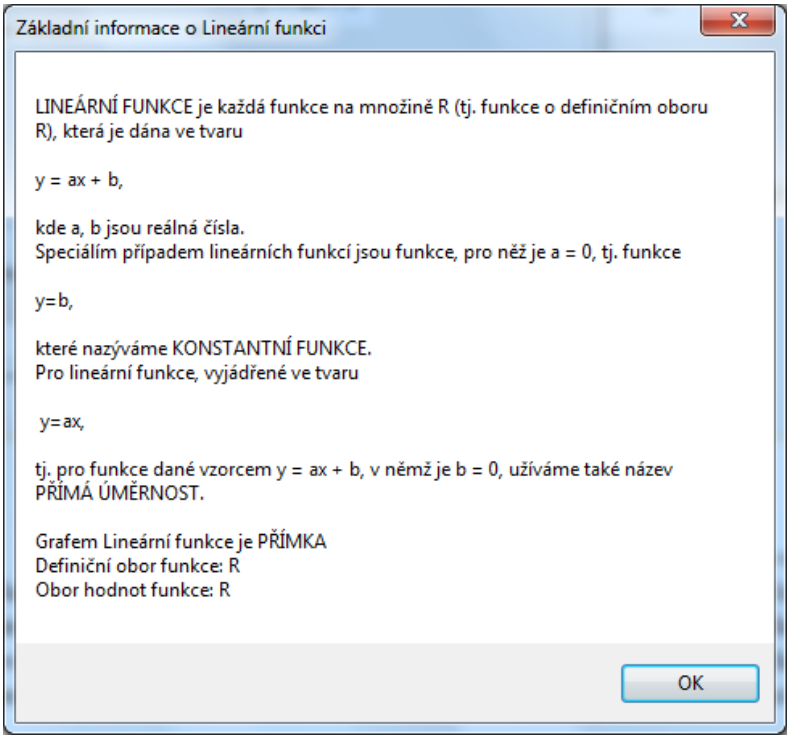
Obrázek 1 - vývojové prostředí

2. Úvodní formulář aplikace



Obrázek 2 - úvodní formulář

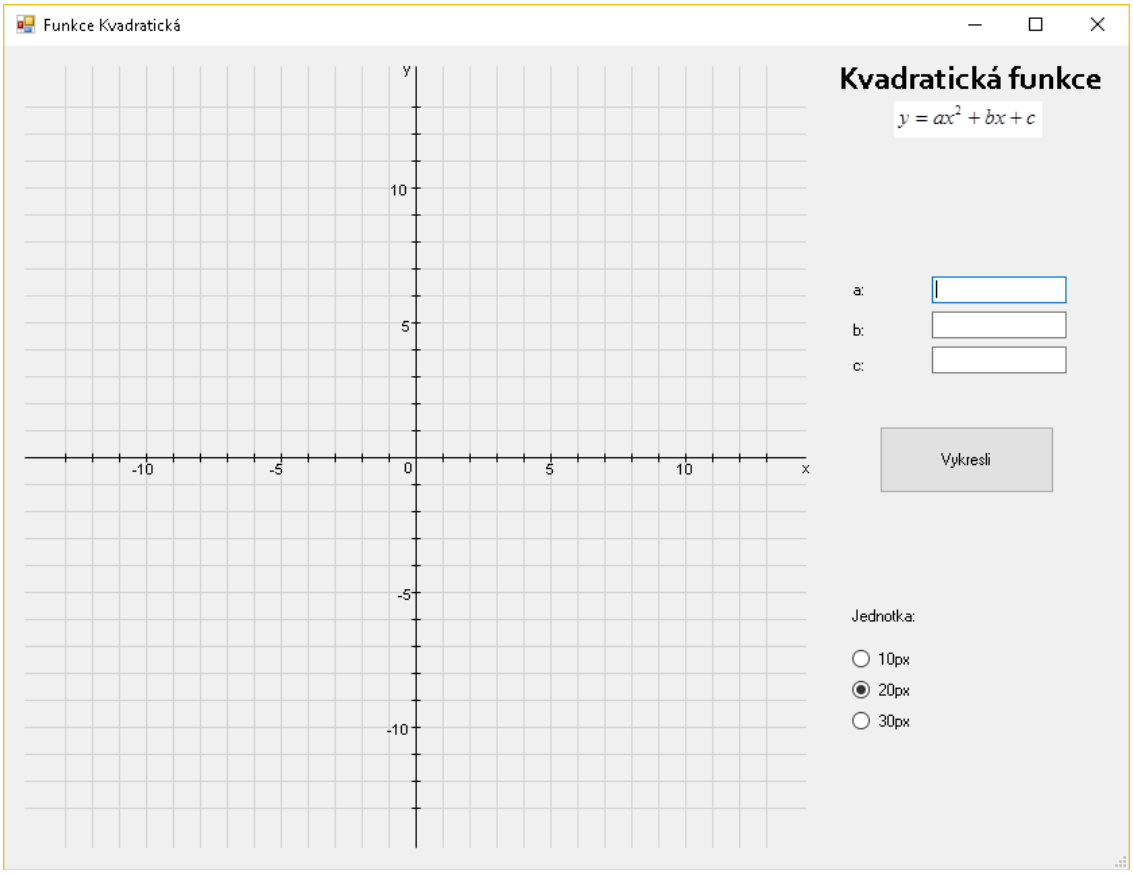
Po spuštění aplikace se objeví úvodní okno programu, které slouží pro výběr dané funkce a získání základních informací o vybrané funkci. Pro pokračování k vykreslování funkce uživatel stiskne tlačítko „Vyber funkci“. Po kliknutí na dané tlačítko se objeví nové okno programu, kde uživatel bude pracovat s předem vybranou funkcí seznamu. Po stisku tlačítka „Info“ se zobrazí okno s informacemi o zvolené funkci.



Obrázek 3 - Základní informace ke zvolené funkci

3. Princip vykreslování grafu

3.1 Soustava souřadná a nastavení jednotky



Obrázek 4 – 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, 20px 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. V podmínkách se zároveň nastaví rozsah hodnot, které budeme dosazovat do proměnné x.

```
double rozsah = 0;
            if (radioButton1.Checked)
            {
                jednotka = 10;
                rozsah = 29;
            }
            if (radioButton2.Checked)
            {
                jednotka = 20;
                rozsah = 1.1;
            }
            if (radioButton3.Checked)
            {
                jednotka = 30;
                rozsah = 9.6;
                   }
```

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 a bylo možné podle těchto parametrů vytvořit seznam s alespoň dvěma body. Pokud ano, tak se tužkou červené barvy vykreslí křivka tvořená body zadanými v poli body[].

```
             if (zadáno)
            {
                tužka.Color = Color.Red;
                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>();
```

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.

Před vykreslením křivky musíme pole body [] nejprve vytvořit ze seznamu bodů, který se vytvoří po stisku tlačítka „Vykresli“.

V první verzi programu jsem se snažil vytvářet rovnou pole. Ale 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.

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 a rozsahu, 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 je potřeba vytvořit seznam bodů. Pokud je graf tvořen více křivkami (např. dvě větve hyperboly u nepřímé úměrnosti nebo jednotlivé křivky u tangens a kotangens), pak musíme vytvořit seznam pro každou křivku. Seznam je datová struktura se složkami stejného typu, které se vzájemně rozlišují indexem. Indexuje se celými čísly od 0. Na rozdíl od pole, které musí mít při deklaraci danou délku, seznam vzniká prázdný a prvky se do něj vkládají metodou Add.

Souřadnice bodů v seznamu 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);
```

Hodnoty xpx a ypx použijeme pro vytvoření bodu a jeho přidání do seznamu. Ještě předtím musíme otestovat, zda nám y „nevyletělo ven z grafu“. Tedy jestli je ypx v rozsahu od 10px do 589px.

```
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 -rozsah. Po dopočítání y podle předpisu funkce a vytvoření bodu v pixelech se x zvětší o 0,2 jednotky. Cyklus pro výpočet se opakuje, dokud je x menší nebo rovno rozsahu.

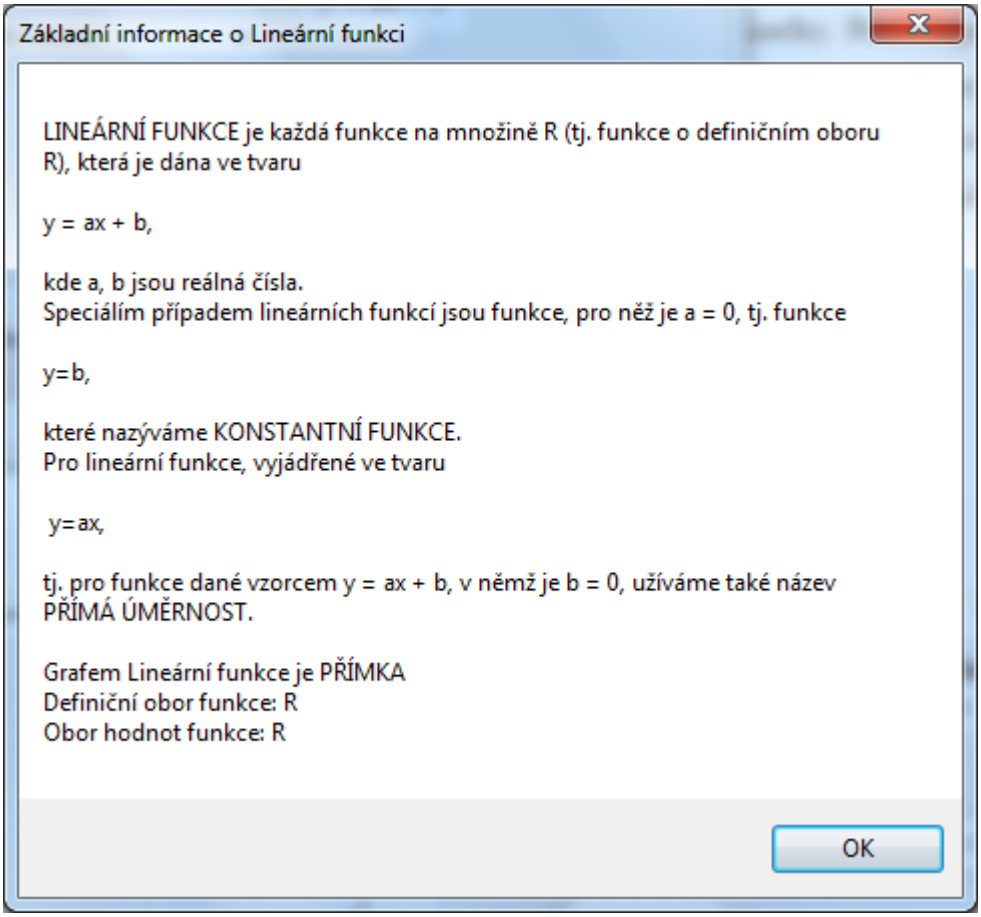
```
double x = -rozsah;
 double y;
 int xpx, ypx;
 while (x <= rozsah)
 {

     double vrcholX = -b / (2 * a);
     for (int index = 0; index <= 16; index++)
     {
         y = a * x * x + b * x + c;
         xpx = (int)(x * jednotka + 300);
         ypx = (int)(-y * jednotka + 300); ;
         x += 0.2;
            }
```

Celý zdrojový kód metody tlačítkoKresliGraf_Click je součástí přílohy.

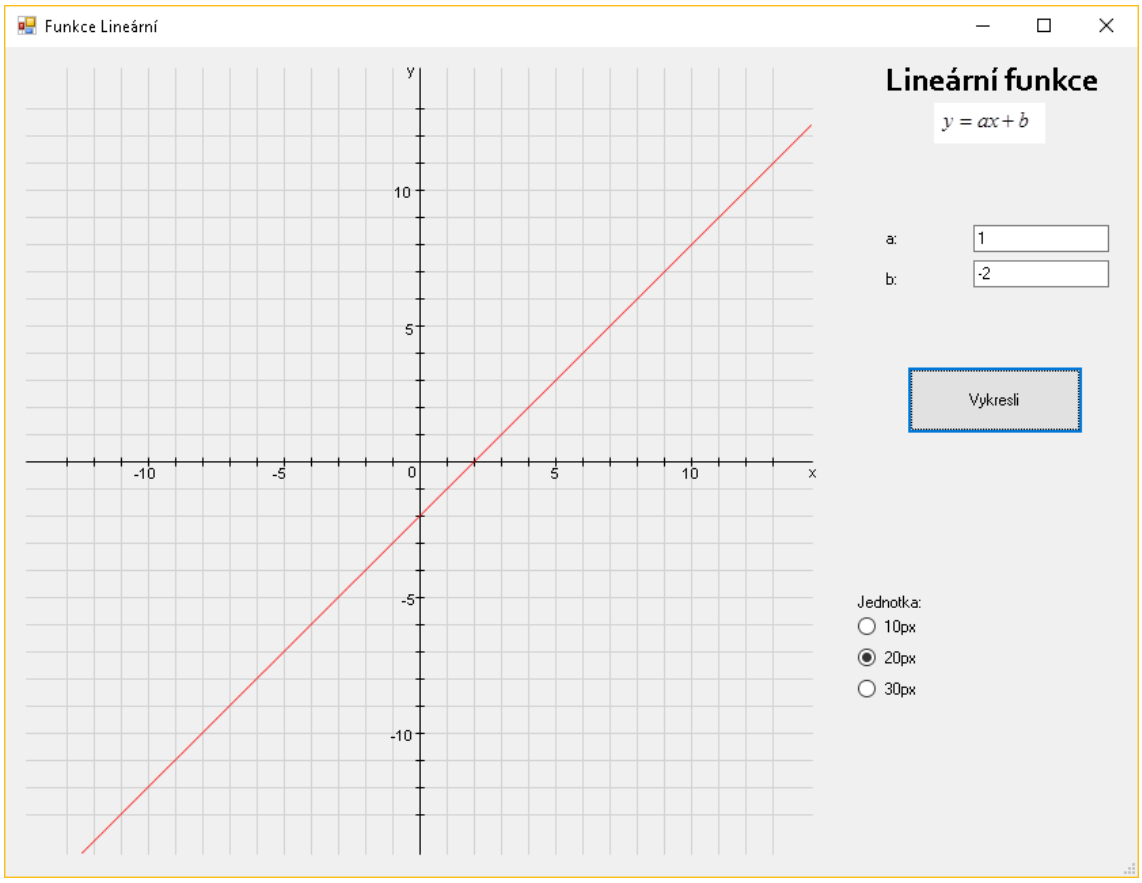
4. Okna jednotlivých funkcí

4.1 Lineární funkce



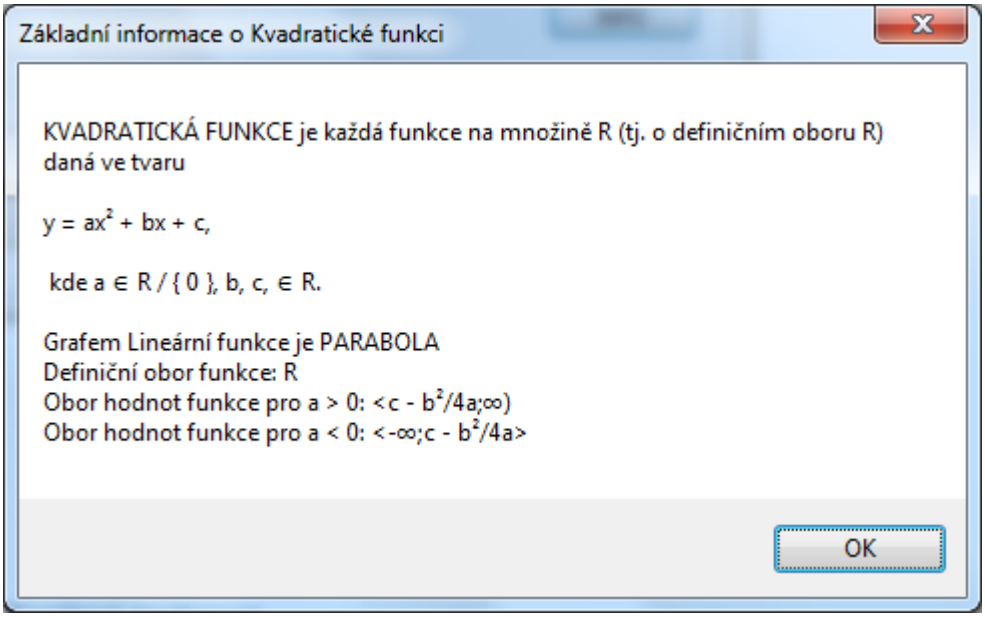
Obrázek 5 - Vlastnosti lineární funkce

U lineární funkce stačí pouze 2 body pro vykreslení úsečky. Proto nepotřebujeme seznam, ale stačí pole se dvěma body. Od hodnoty x=-rozsah se snažíme vytvořit první bod. Pokud pro x=-rozsah není bod ve vykreslované oblasti grafu, tak zvětšujeme x o 0,2 tak dlouho, dokud se do dané oblasti nedostaneme. Druhý bod vzniká stejně z druhé strany. Tehdy x = rozsah a zmenšujeme.

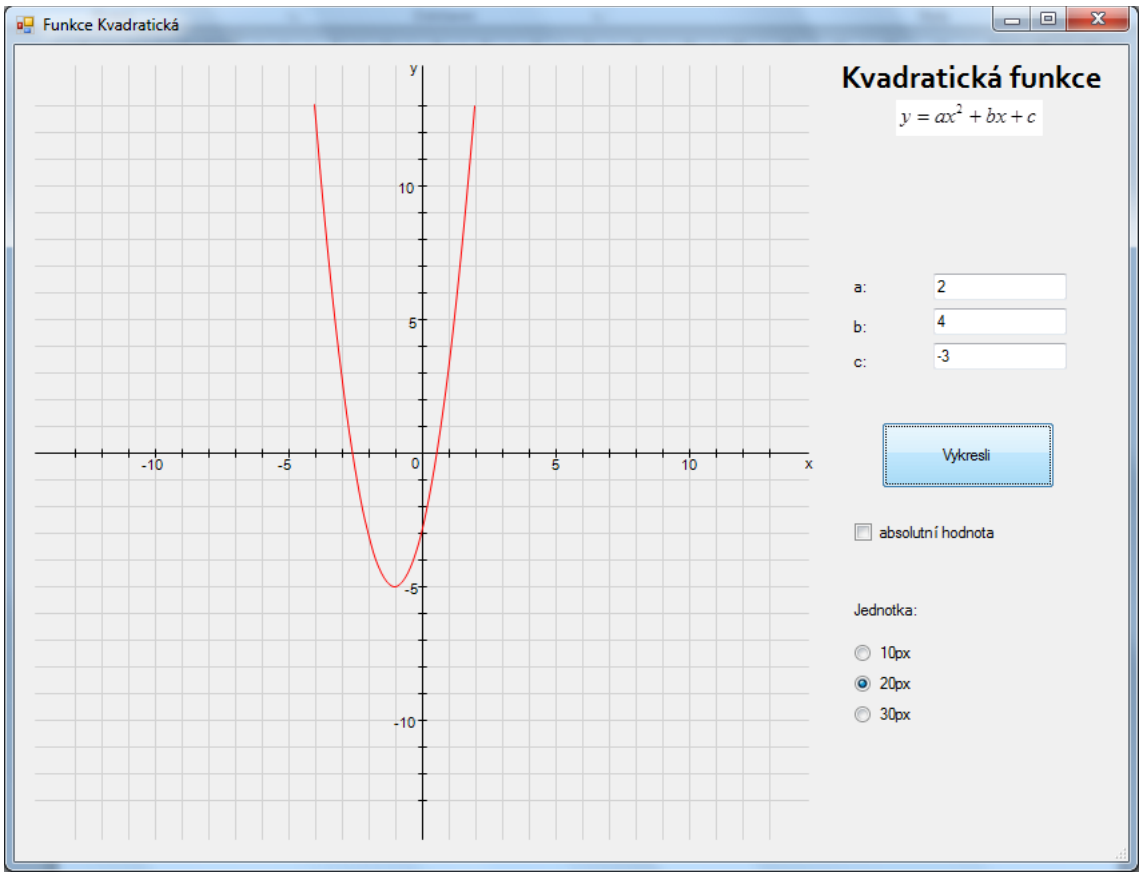


Obrázek 6 - graf lineární funkce

4.2 Kvadratická funkce

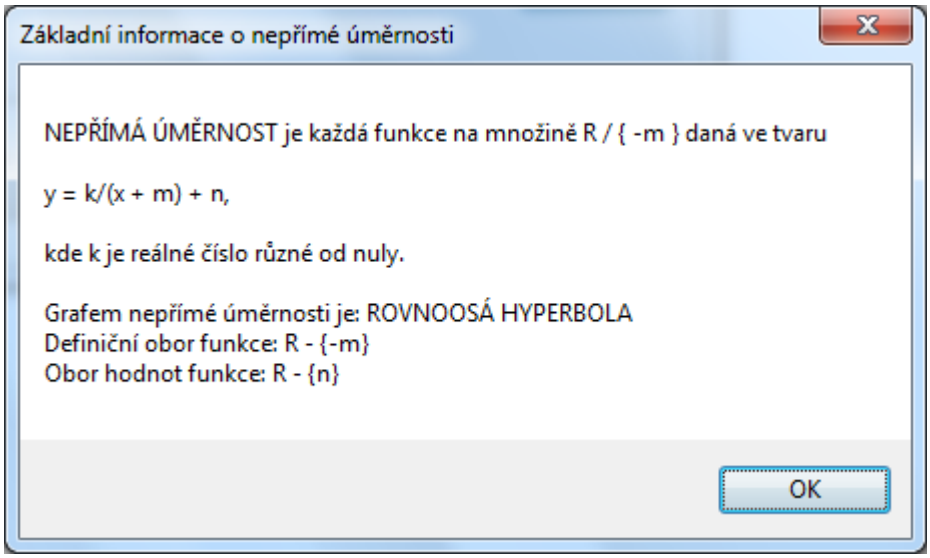


Obrázek 7 - Vlastnosti kvadratické funkce

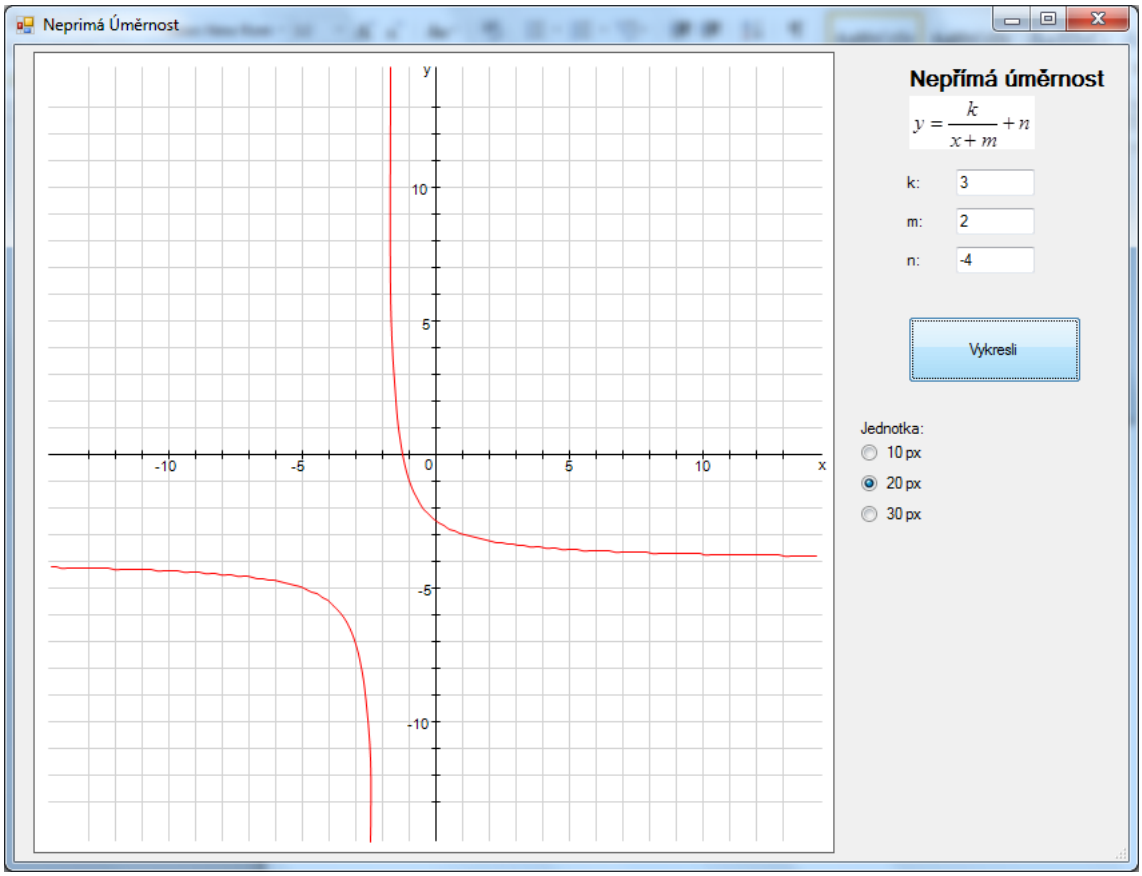


Obrázek 8 - graf kvadratické funkce

4.3 Nepřímá úměrnost



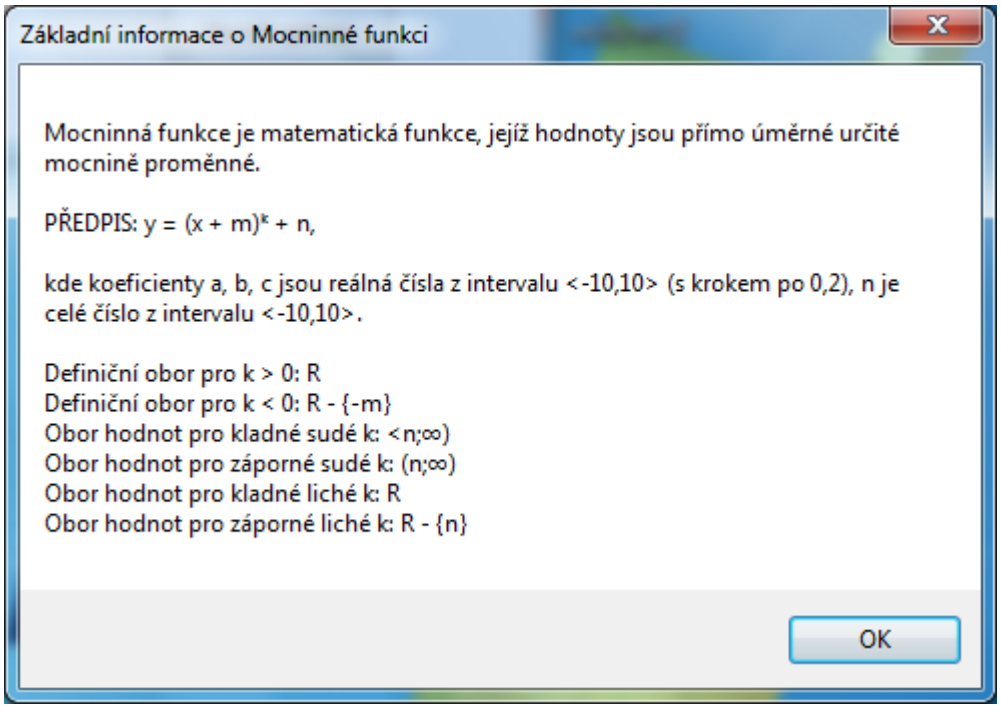
Obrázek 9 - Vlastnosti nepřímé úměrnosti



Obrázek 10 - graf nepřímé úměrnosti

Aby se obě křivky vykreslovaly s náznakem nekonečna, je pro každou křivku v poli bodů přidán jeden bod. Pro levou křivku poslední bod a pro pravou první.

4.4 Mocninná funkce

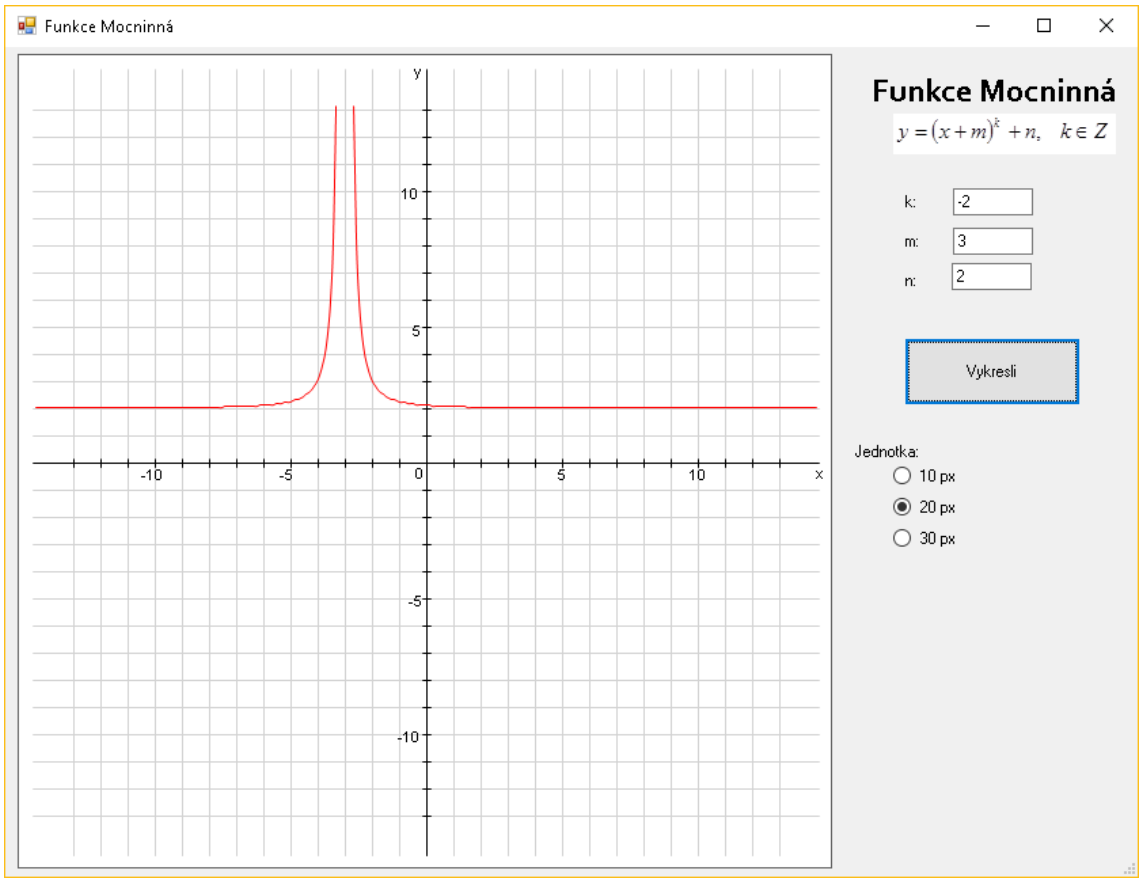


Obrázek 11 - Vlastnosti mocninné funkce

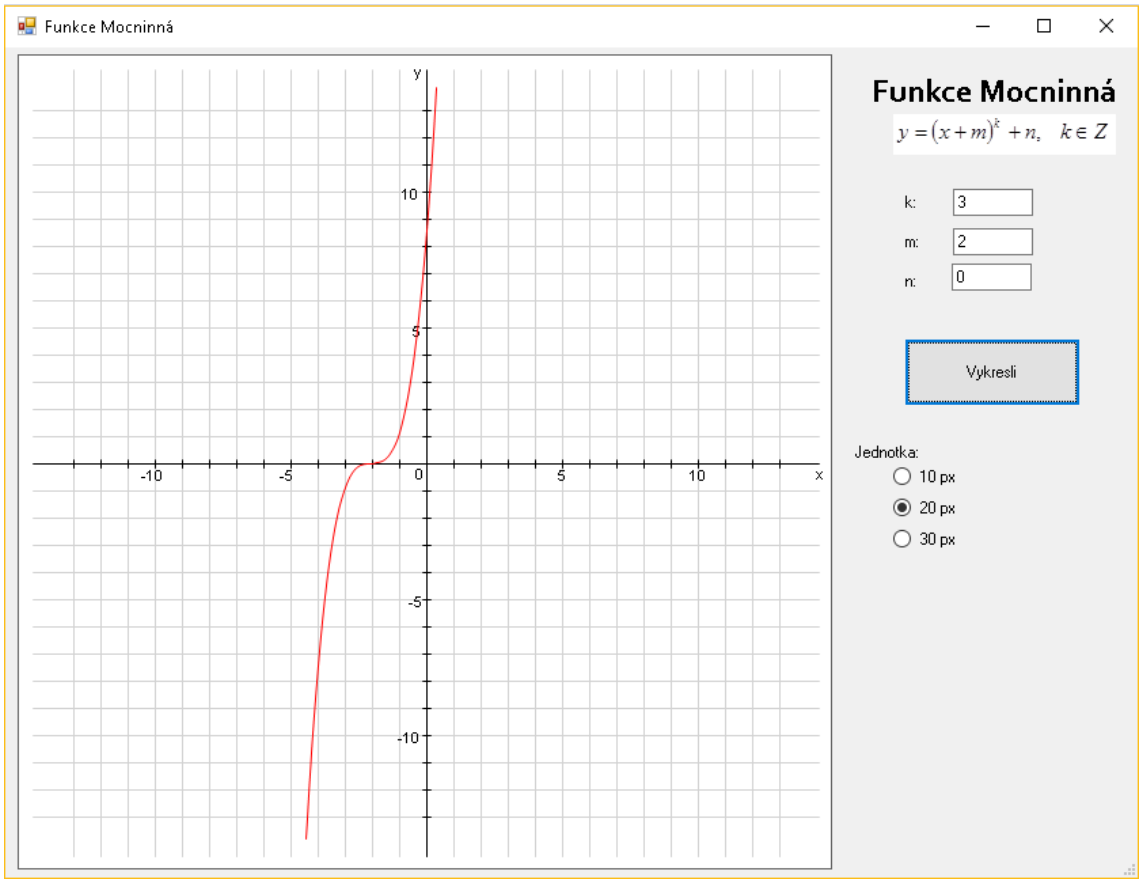
Pro k>0 je grafem funkce jedna křivka, při jejím vykreslení postupujeme stejně jako u kvadratické funkce. Pro k<0 tvoří graf funkce dvě křivky, při jejich vykreslení postupujeme stejně jako u nepřímé úměrnosti.

```
if (zadáno)
            {
                tužka.Color = Color.Red;
                if (k > 0)
                {
                    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>();
                }
                if (k < 0)
                {
                    int početBodů1 = seznamBodů1.Count();
                    Point[] body1 = new Point[početBodů1];
                    for (int index = 0; index < početBodů1; index++)
                        body1[index] = seznamBodů1[index];
                    kp.DrawCurve(tužka, body1);
                    seznamBodů1 = new List<Point>();

                    int početBodů2 = seznamBodů2.Count();
                    Point[] body2 = new Point[početBodů2];
                    for (int index = 0; index < početBodů2; index++)
                        body2[index] = seznamBodů2[index];
                    kp.DrawCurve(tužka, body2);
                    seznamBodů2 = new List<Point>();
                }
```

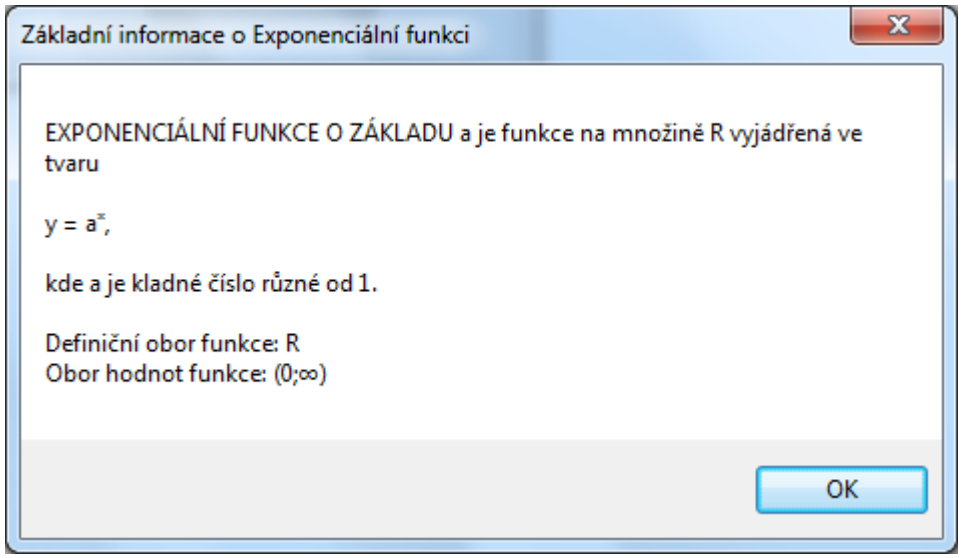


Obrázek 12 - graf mocninné funkce pro k>0

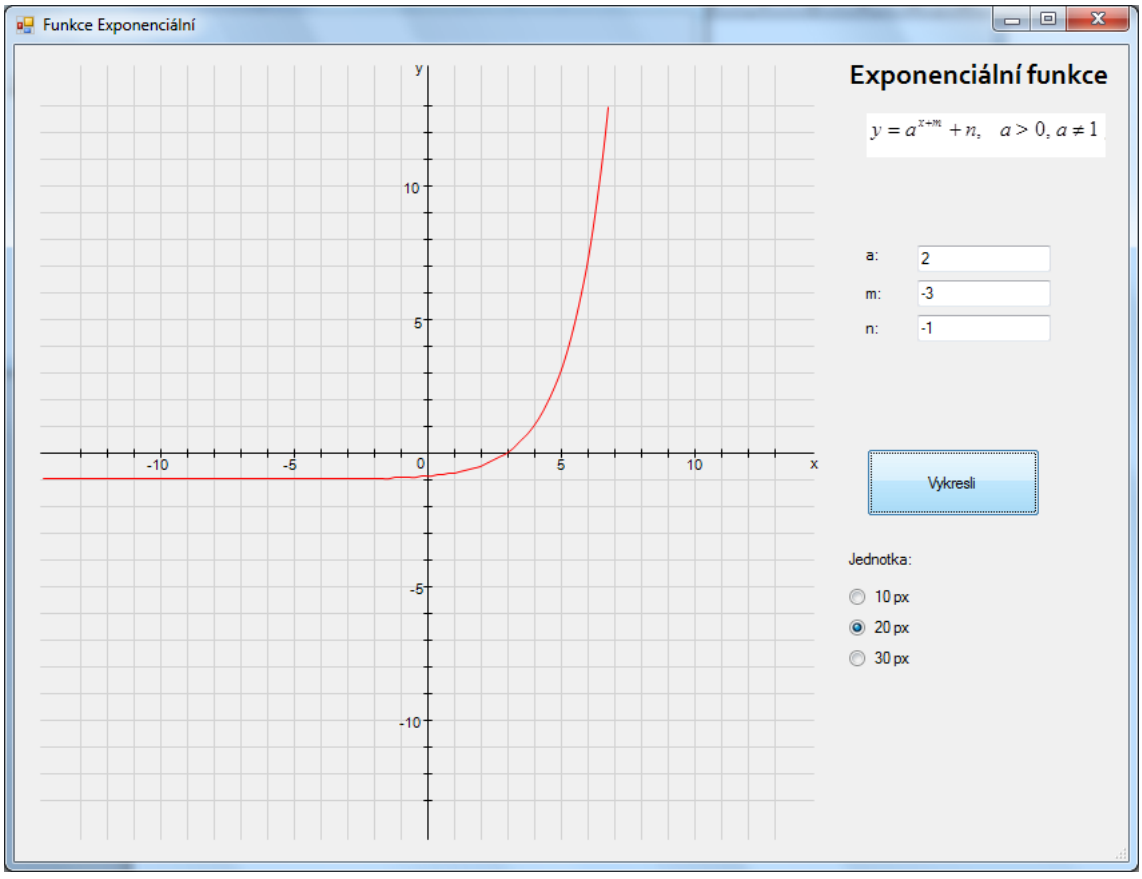


Obrázek 13 - graf mocninné funkce pro k<0

4.5 Exponenciální funkce

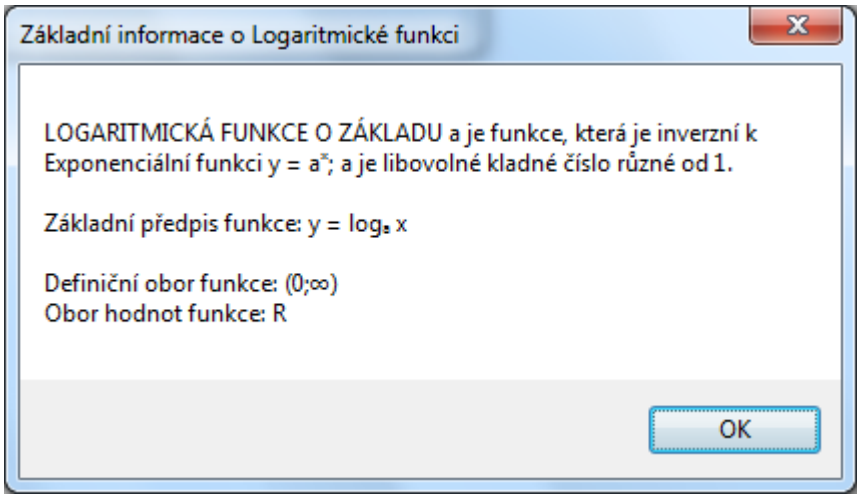


Obrázek 14 - Vlastnosti exponenciální funkce

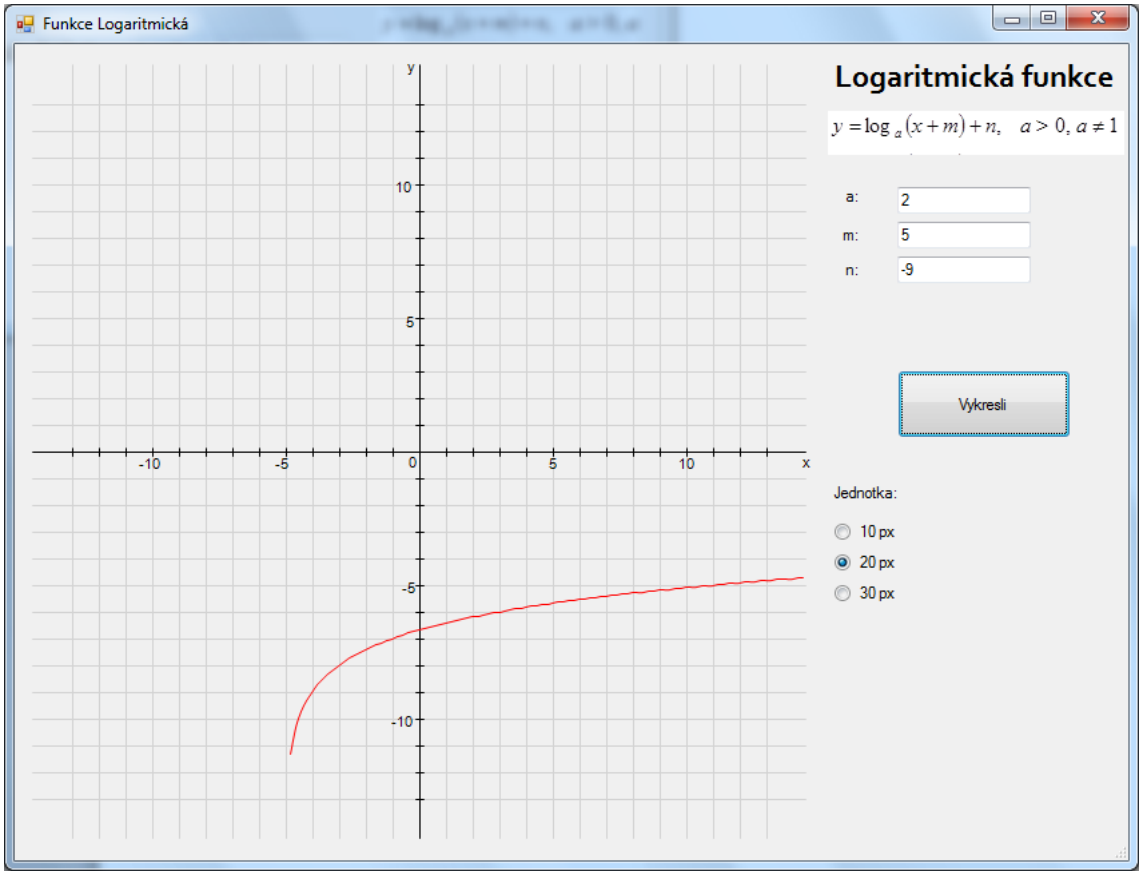


Obrázek 15 - graf exponenciální funkce

4.6 Logaritmická funkce

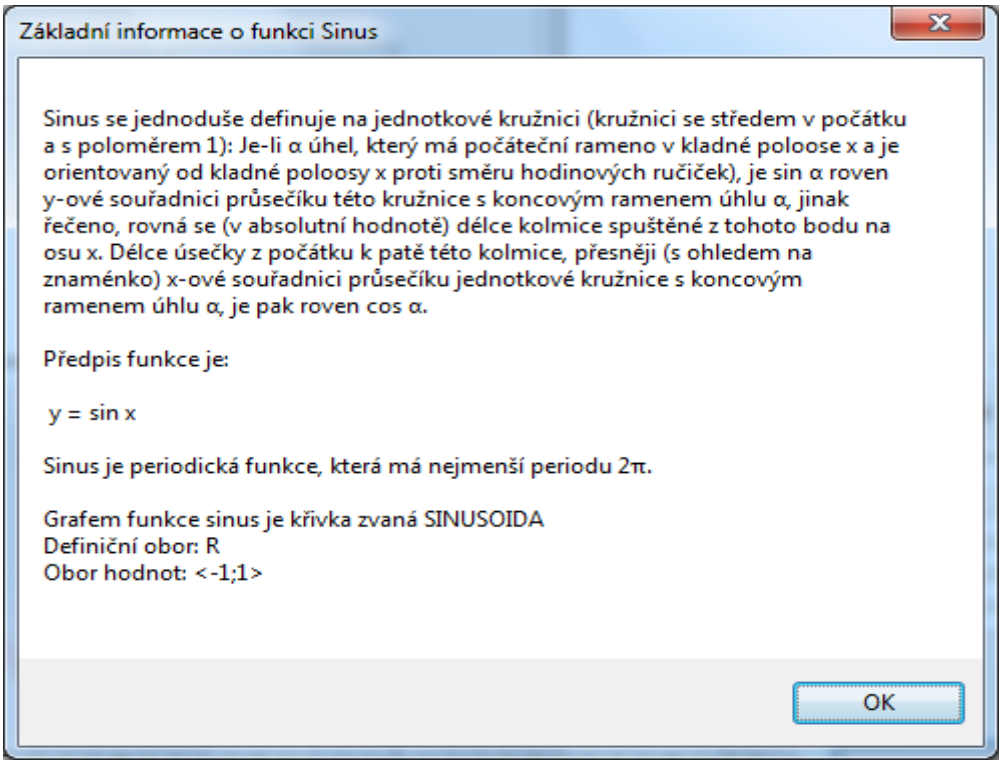


Obrázek 16 - Vlastnosti logaritmické funkce

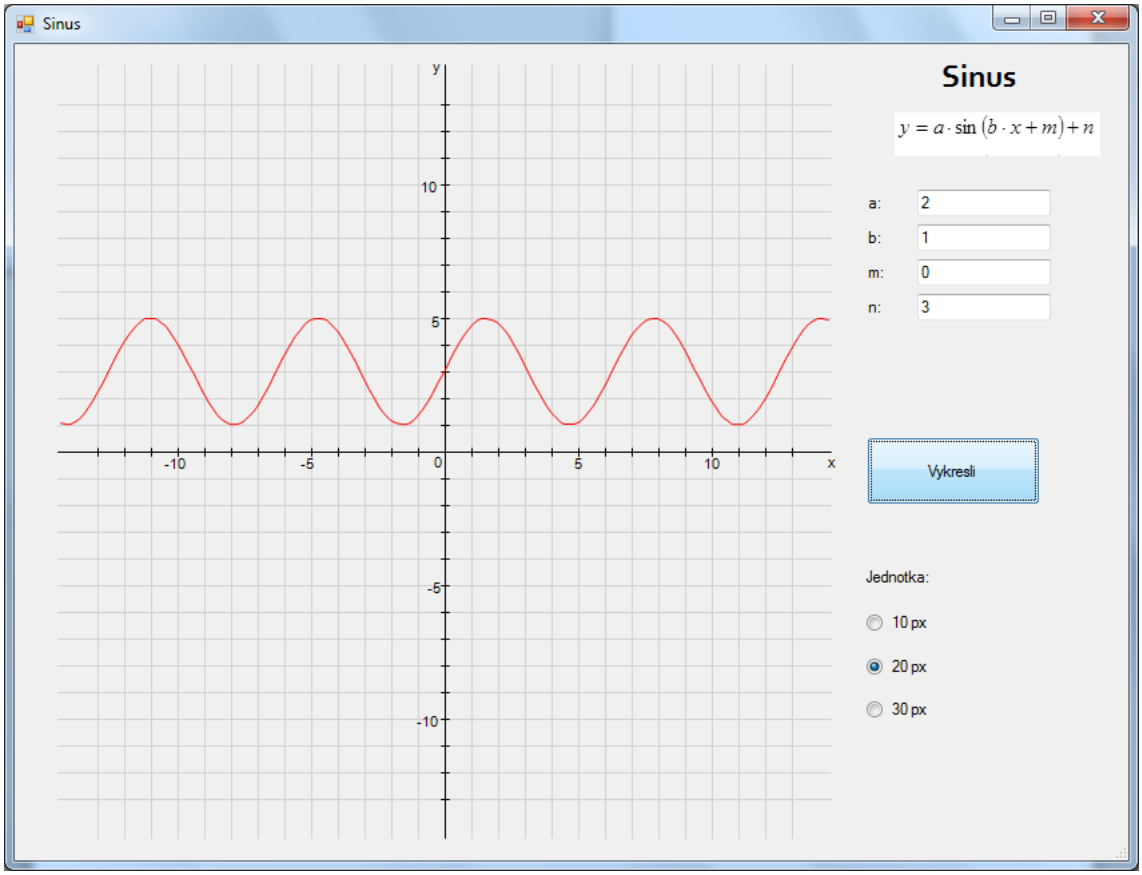


Obrázek 17 - graf logaritmické funkce

4.7 Sinus

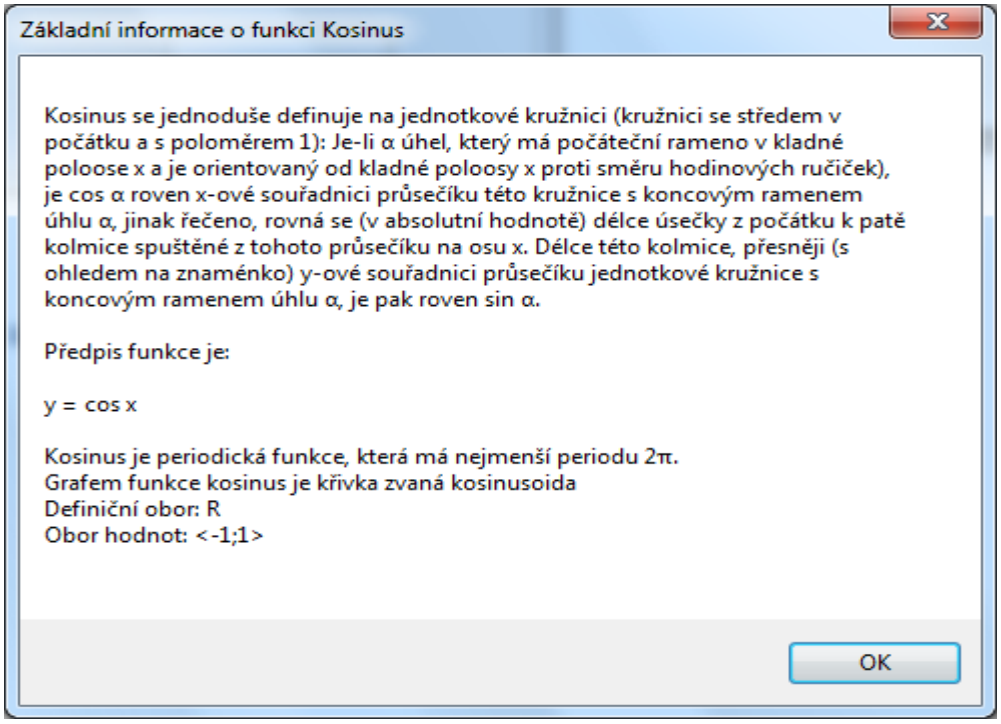


Obrázek 18 - Vlastnosti funkce sinus

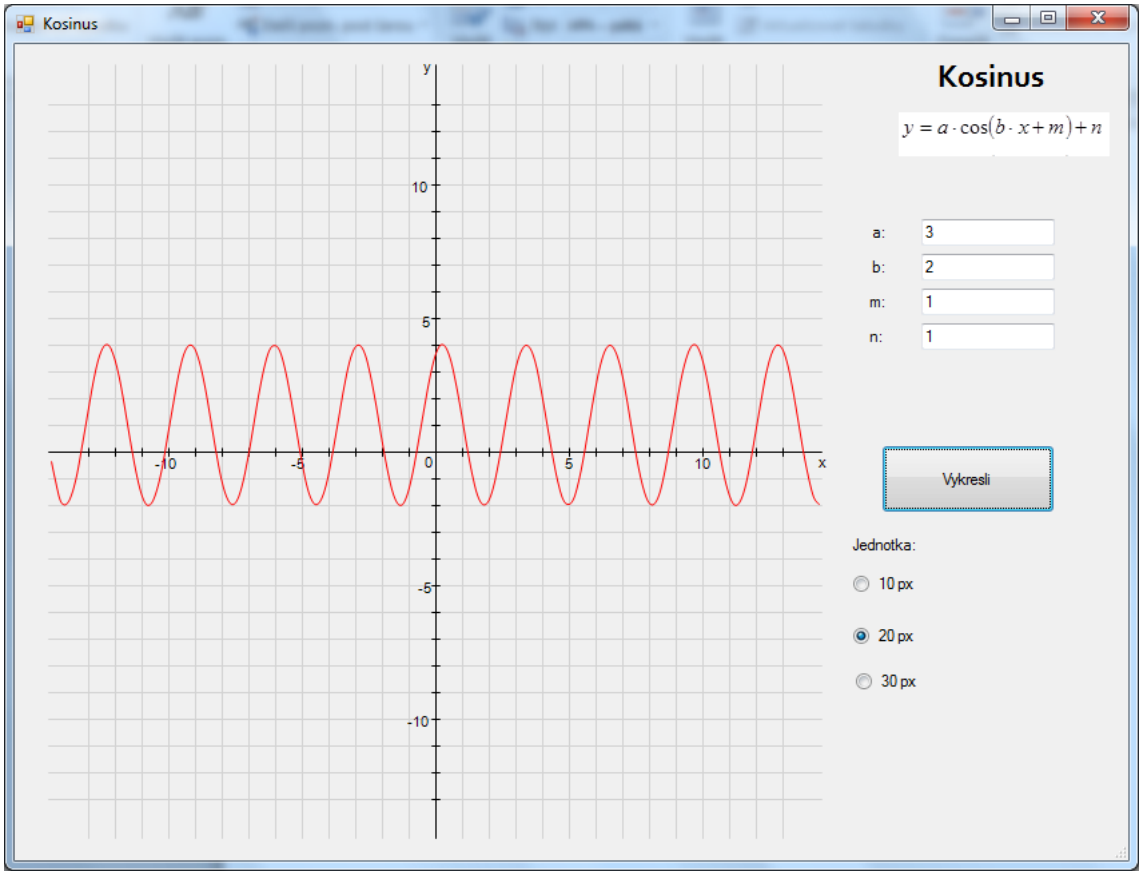


Obrázek 19 - graf funkce sinus

4.8 Kosinus

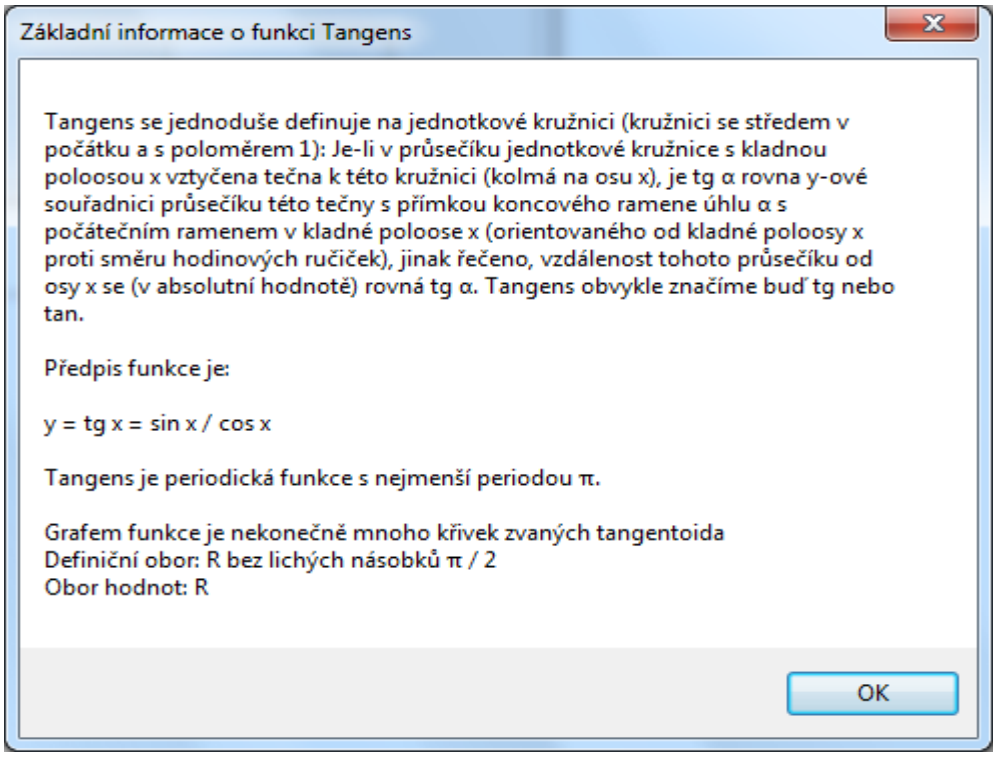


Obrázek 2012 - Vlastnosti funkce kosinus

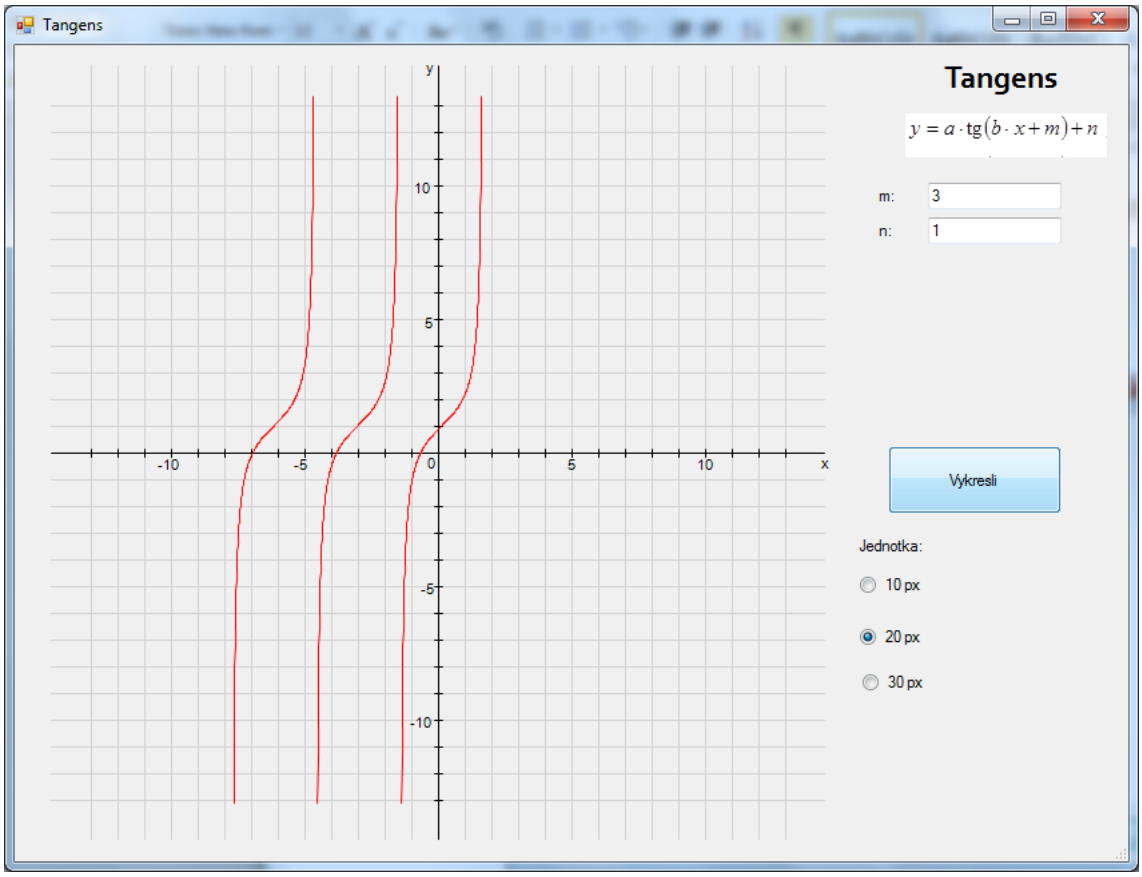


Obrázek 21 - graf funkce kosinus

4.9 Tangens

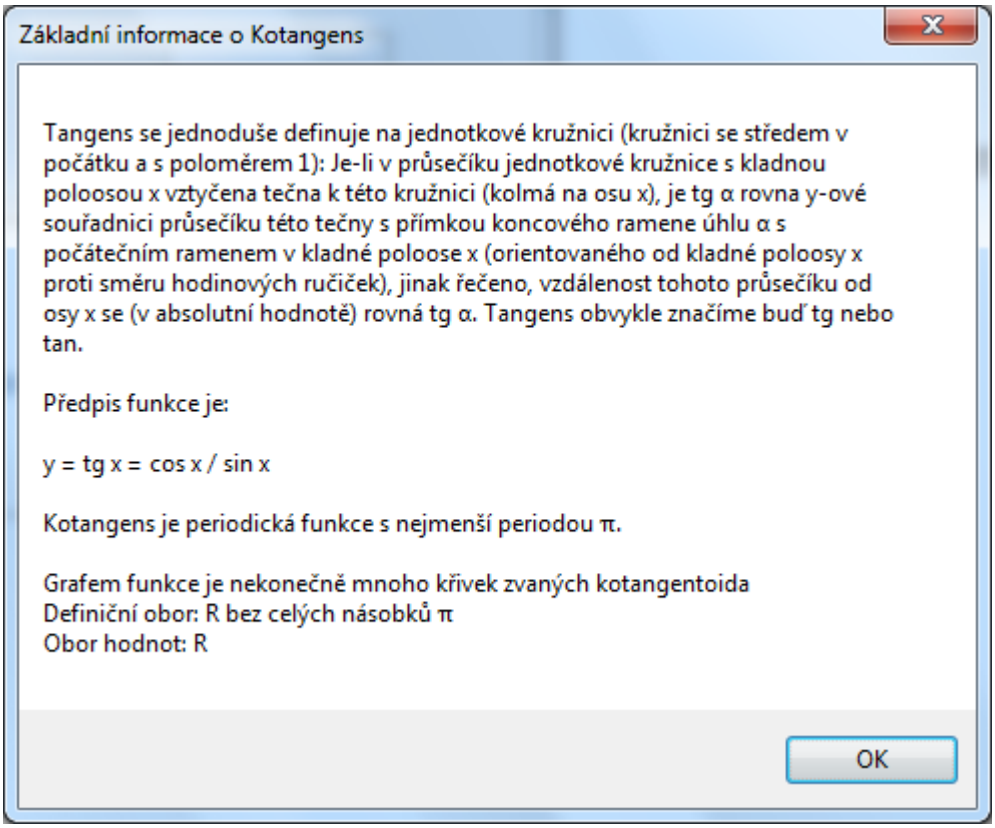


Obrázek 22 - Vlastnosti funkce tangens

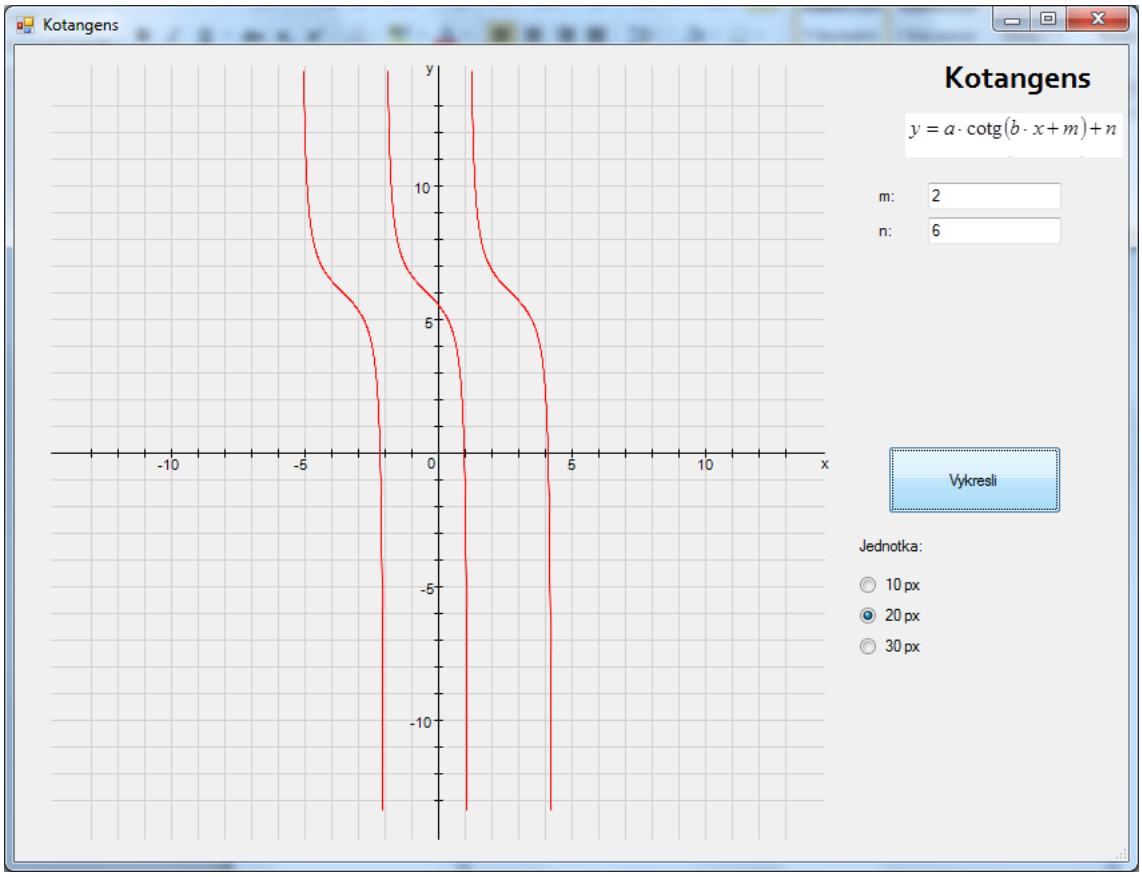


Obrázek 23 - graf funkce tangens

4.10 Kotangens



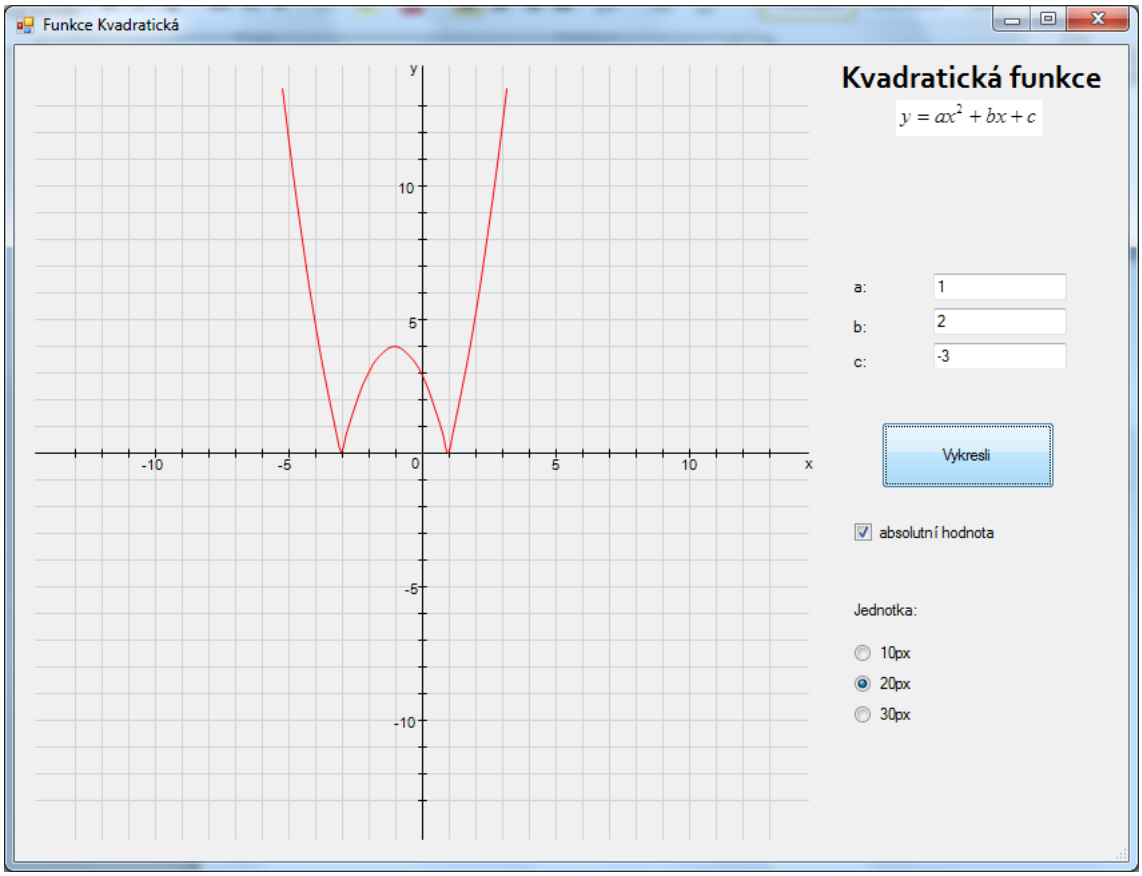
Obrázek 24 - Vlastnosti funkce kosinus



Obrázek 25- graf funkce kotangens

4.11 Absolutní hodnota

Do formuláře pro vykreslování Kvadratické funkce jsem přidal zaškrtávací pole, které umožnuje nastavit použití absolutní hodnoty. Je-li pole zaškrtnuté, pak se před vykreslením křivky u všech bodů, které jsou pod osou x, změní souřadnice y tak, aby se daný bod otočil kolem osy x.



Obrázek 2613 - graf kvadratické funkce s absolutní hodnotou

5. Debugging

Při psaní této aplikace jsem narazil na spoustu problému, které byly schopné přerušit chod programu. Jeden z těchto problémů byl pád aplikace poté, co se graf funkce vykreslil mimo vykreslovací panel. Pomocí funkce if jsem nastavil podmínku, která zobrazí chybovou hlášku poté, co se graf funkce vykreslí mimo vykreslovací panel. Díky této podmínce se vyřeší další problém. Tento problém spočíval v pádu aplikace po shození na lištu a zpětném obnovení z lišty. Chyba byla v tom, že po shození programu se vyprázdnil seznam bodů, ze kterého se funkce vykreslovala. Proto po zpětném načtení programu aplikace spadlo, protože neměla žádné body na vykreslování. Další problém nastal u funkcí Sinus a Kosinus. U těchto funkcí nastával problém s deformací křivky po zadání hodnot menší jak 15 a menší jak -15 do pole a. Problém jsem vyřešil omezením zadání hodnot do PoleA na reálná číslo od -14 do 14.

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.

Provedl jsem všechna vylepšení předchozí verze, která jsem plánoval.

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

Příloha 1

Zdrojový kód pro okno „Mocninná funkce“

```
public partial class MocninnáFunkce : Form
    {
        double m = 0;
        double n = 0;
        int k;
        bool zadáno = false;
        List<Point> seznamBodů = new List<Point>();
        List<Point> seznamBodů1 = new List<Point>();
        List<Point> seznamBodů2 = new List<Point>();

        int jednotka = 20;

        public MocninnáFunkce()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            double rozsah = 0;
            if (radioButton1.Checked)
            {
                jednotka = 10;
                rozsah = 29;
            }
            if (radioButton2.Checked)
            {
                jednotka = 20;
                rozsah = 14.4;
            }
            if (radioButton3.Checked)
            {
                jednotka = 30;
                rozsah = 9.6;
            }
            try
            {
                k = Convert.ToInt32(PoleK.Text);
                m = Convert.ToDouble(PoleM.Text);
                n = Convert.ToDouble(PoleN.Text);
            }
            catch
            {
                MessageBox.Show("Chybně zadaná vstupní data", "CHYBA!");
                return;
            }
            if (k == 0)
                MessageBox.Show("Chybně zadaná vstupní data", "CHYBA!");
            else
            {
                if (k > 0)
                {

                    double x = -rozsah;
                    double y;
                    int xpx, ypx;
                    while (x <= rozsah)
```

```
            {
                y = Math.Pow((x + m), k) + n;
                xpx = (int)(x * jednotka + 300);
                ypx = (int)(-y * jednotka + 300);
                if ((ypx >= 10) && (ypx <= 589))
                {
                    Point bod = new Point(xpx, ypx);
                    seznamBodů.Add(bod);
                }
                x += 0.2;
            }
        }

        if (k < 0)
        {
            double x1 = -rozsah;
            double y1;
            int xpx1, ypx1;
            while (x1 < -m)
            {
                y1 = Math.Pow((x1 + m), k) + n;
                xpx1 = (int)(x1 * jednotka + 300);
                ypx1 = (int)(-y1 * jednotka + 300);
                if ((ypx1 >= 10) && (ypx1 <= 589))
                {
                    Point bod1 = new Point(xpx1, ypx1);
                    seznamBodů1.Add(bod1);
                }
                x1 += 0.1;
            }


            double x2 = -m + 0.2;
            double y2;
            int xpx2, ypx2;
            while (x2 <= rozsah)
            {
                y2 = Math.Pow((x2 + m), k) + n;
                xpx2 = (int)(x2 * jednotka + 300);
                ypx2 = (int)(-y2 * jednotka + 300);
                if ((ypx2 >= 10) && (ypx2 <= 589))
                {
                    Point bod2 = new Point(xpx2, ypx2);
                    seznamBodů2.Add(bod2);
                }
                x2 += 0.1;
            }
        }

        zadáno = true;
        panel1.Refresh();
    }
}

private void panel1_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 * jed-
notka, 589);
                kp.DrawLine(tužka, 300 - index * jednotka, 10, 300 - index * jed-
notka, 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);
            if (jednotka > 10)           //jednotka 20 a 30
            {
                kp.DrawString("-5", new Font("Arial", 8), Brushes.Black, 290 - 5
* jednotka, 302); //číslo -5 na X
                kp.DrawString("5", new Font("Arial", 8), Brushes.Black, 295 + 5 *
jednotka, 302); //číslo 5 na X
                kp.DrawString("-5", new Font("Arial", 8), Brushes.Black, 285, 295
+ 5 * jednotka); //číslo -5 na Y
                kp.DrawString("5", new Font("Arial", 8), Brushes.Black, 288, 296
- 5 * jednotka); //číslo 5 na Y
            }
            if (jednotka < 30)            //jednotka 10 a 20
            {
                kp.DrawString("-10", new Font("Arial", 8), Brushes.Black, 288 -
10 * jednotka, 302); //číslo -10 na X
                kp.DrawString("10", new Font("Arial", 8), Brushes.Black, 292 + 10
* jednotka, 302); //číslo 10 na X
                kp.DrawString("-10", new Font("Arial", 8), Brushes.Black, 277,
295 + 10 * jednotka); //číslo -10 na Y
                kp.DrawString("10", new Font("Arial", 8), Brushes.Black, 280, 295
- 10 * jednotka); //číslo 10 na Y
            }
            if (jednotka == 10)
            {
                kp.DrawString("-20", new Font("Arial", 8), Brushes.Black, 285 -
20 * jednotka, 302); //číslo -10 na X
                kp.DrawString("20", new Font("Arial", 8), Brushes.Black, 295 + 20
* jednotka, 302); //číslo 10 na X
                kp.DrawString("-20", new Font("Arial", 8), Brushes.Black, 280,
295 + 20 * jednotka); //číslo -10 na Y
                kp.DrawString("20", new Font("Arial", 8), Brushes.Black, 280, 295
- 20 * 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;
        if (k > 0)
        {
            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>();
        }

        if (k < 0)
        {
            int početBodů1 = seznamBodů1.Count();
            Point[] body1 = new Point[početBodů1];
            for (int index = 0; index < početBodů1; index++)
                body1[index] = seznamBodů1[index];
            kp.DrawCurve(tužka, body1);
            seznamBodů1 = new List<Point>();

            int početBodů2 = seznamBodů2.Count();
            Point[] body2 = new Point[početBodů2];
            for (int index = 0; index < početBodů2; index++)
                body2[index] = seznamBodů2[index];
            kp.DrawCurve(tužka, body2);
            seznamBodů2 = new List<Point>();
        }
    }
}
```

Příloha 2

Obsah vloženého CD

- Text práce ve formátu .pdf
- Složka s kompletním zdrojovým kódem aplikace
- Samostatné 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