JEDNODUCHÉ PRÍKLADY POUŽITIA SYSTÉMU WEB*MATHEMATICA* PRE UČITEĽOV

V tejto kapitole si ukážeme niekoľko ďalších príkladov, ktoré rozšíria naše schopnosti používať web*MATHEMATICU* pri vytváraní jsp aplikácií. Na rozdiel od predchádzajúcej kapitoly sa už podrobne nebudeme venovať otázkam odovzdávania hodnôt premenných v rámci formulára, interpretácii premenných. Predpokladáme tiež, že čitateľ už všetky príkazy web*MATHEMATICE* uvedené v predchádzajúcej kapitole pozná a dokáže vhodne používať.

Príklady do tejto kapitoly sme volili s ohľadom na užívateľov tak, aby s ich pomocou bolo možné vytvárať čo najširšie spektrum výučbových aplikácií pre študentov. Budeme sa venovať jednak otázkam grafiky a zobrazovania zložitejších grafických výsledkov na stránke, ďalej otázkam práce s funkciami, možnostiam použitia pripravených výpočtových balíkov a kombinácii uvedených techník s MathML kódovaním tak, aby pripravené stránky mali aj pedagogickú hodnotu.

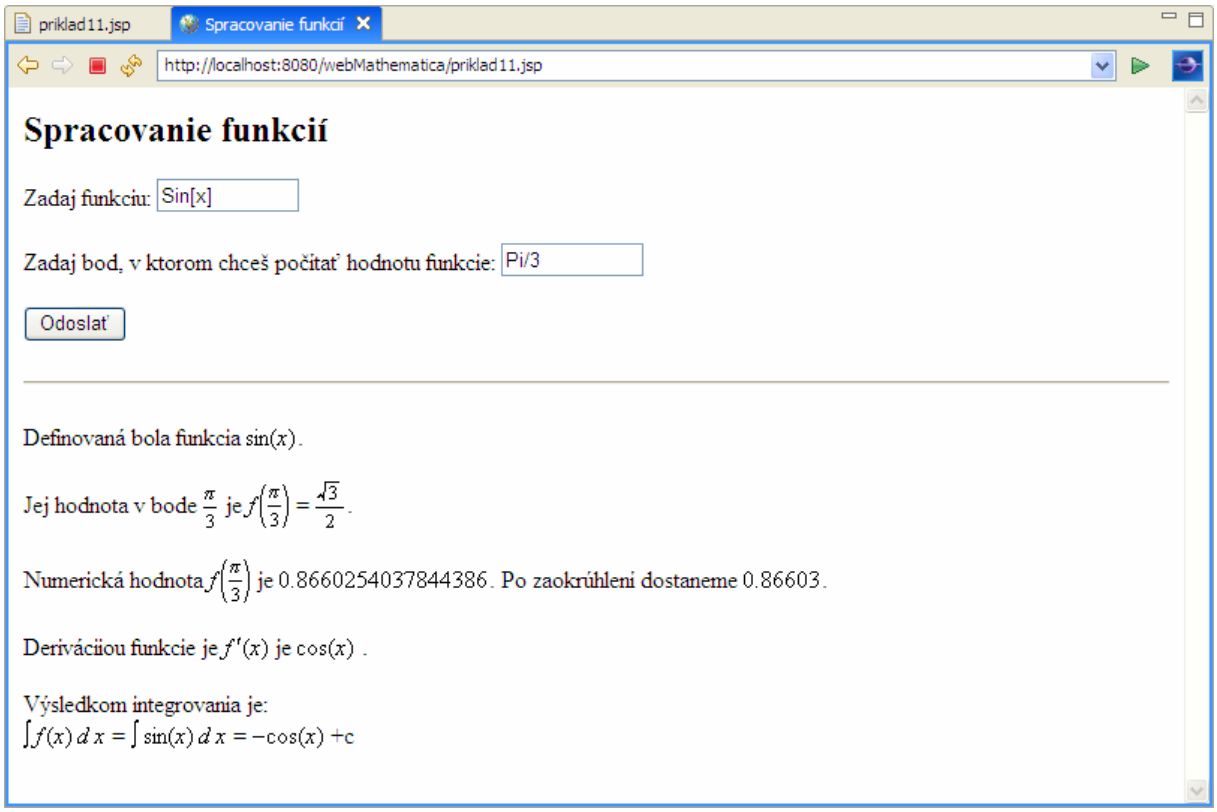
1. Spracovanie funkcií

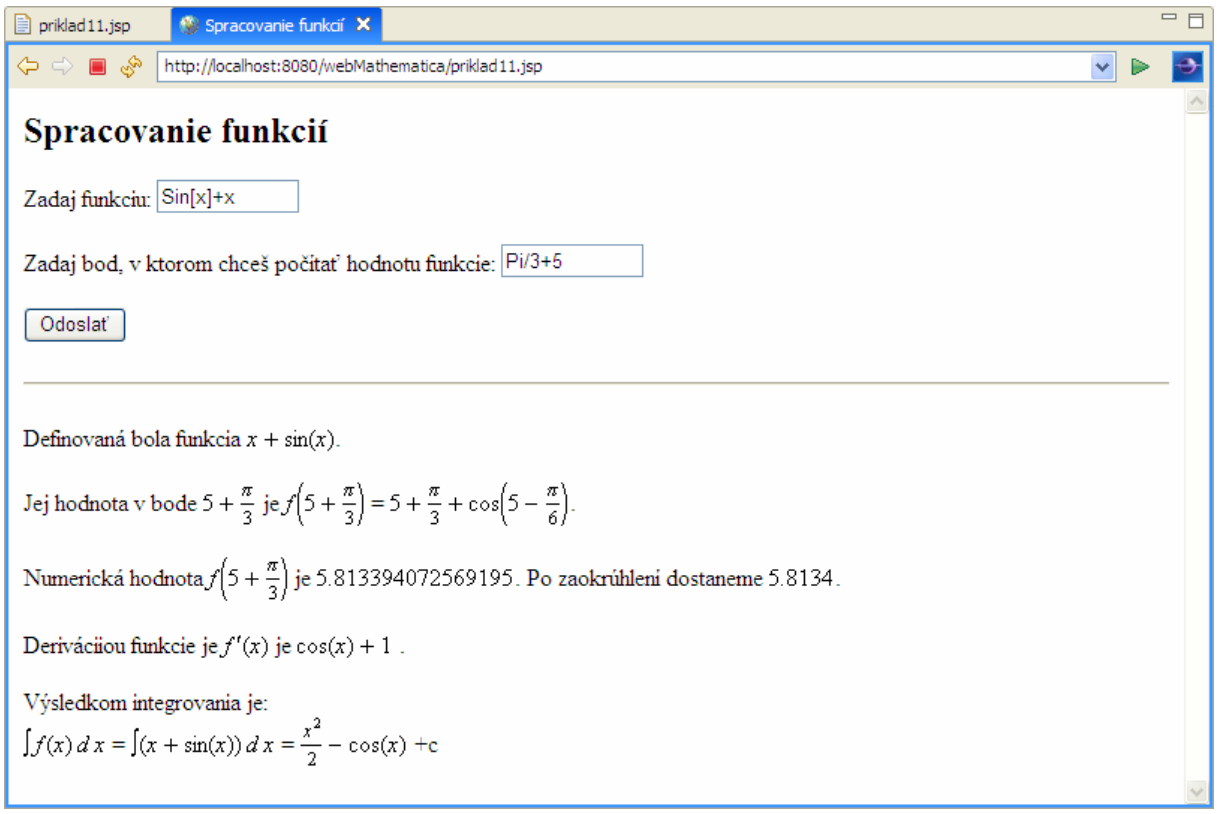
Pri príprave pedagogických materiálov sa veľmi často stretávame s úlohami, v ktorých potrebujeme pracovať s funkciami – nakresliť ich graf, vypočítať deriváciu, integrál a pod. Definovať funkciu nám umožní oveľa jednoduchšiu prácu pri symbolických výpočtoch, ako práca len so samotnými premennými (v zmysle klasického programovania). V nasledujúcej ukážke si uvedieme, ako je možné pracovať s funkciami, zobrazovať výsledky výpočtu a používať funkciu **`HoldForm`** – z *MATHEMATICE*.

Nasledujúci príklad bude slúžiť ako ilustrácia riešenia nasledujúceho problému:

- definovať funkciu a bod, v ktorom budeme počítať napr. hodnotu funkcie
- vypísať s akou funkciou pracujeme
- vypísať hodnotu funkcie aj s komentárom
- vypísať numerickú hodnotu funkcie
- vypísať deriváciu funkcie
- vypísať výpočet a výsledok výpočtu integrálu.

Najskôr uvedieme dve ukážky činnosti takejto jsp stránky a potom jej zdrojový kód.





Teraz sa poďme podrobnejšie pozrieť na zdrojový kód stránky a vysvetlime jeho činnosť.

```
<?xml version="1.0"?>
```

```
<?xml-stylesheet type="text/xsl"
href="/webMathematica/Resources/XSL/mathml.xsl"?>

<%@ page language="java"  %>
<%@ page contentType="text/html;charset=utf-8" %>
<%@ taglib uri="/webMathematica-taglib" prefix="msp" %>

<html xmlns="http://www.w3.org/1999/xhtml">
      <head>
            <meta name="author" content="Monika Kovacova"/>
<meta name="copyright" content="Monika Kovacova, mathematica@mathematica.sk"/>
            <meta http-equiv="Content-type" content="text/html;
charset=utf-8" />
            <title>Spracovanie funkcií</title>
      </head>

<body bgcolor="#ffffff">
<h2>Spracovanie funkcií</h2>

<msp:allocateKernel>
      <msp:evaluate>
         MSPPageOptions[ "ContentType" -> "text/xml"];
      </msp:evaluate>

<form action="priklad11.jsp" method="post">

<p>Zadaj funkciu:
<input type="text" name="fun" size="10"
      value="<msp:evaluate>MSPValue[$$fun,"Sin[x]"]</msp:evaluate>" />
<msp:evaluate>
If[MSPValueQ[$$fun],
      f[x_] = ToExpression[$$fun], f[x_] = Sin[x]];
</msp:evaluate>
</p>

<p>Zadaj bod, v ktorom chceš počítať hodnotu funkcie:
<input type="text" name="bod" size="10"
      value="<msp:evaluate>MSPValue[$$bod,"Pi/3"]</msp:evaluate>" />
<msp:evaluate>
If[MSPValueQ[$$bod],
      bod = ToExpression[$$bod], bod = Pi/3];
</msp:evaluate>
</p>

<input type="submit" name="tlacidlo" value="Odoslať" />
</form>

<hr/>

<p>
<msp:evaluate>
      "Definovaná bola funkcia " <> MSPFormat[ f[x], MathMLForm]
</msp:evaluate>.
</p>

<p>
Jej hodnota v bode
<msp:evaluate> MSPFormat[ bod, MathMLForm] </msp:evaluate> je
<msp:evaluate>
      MSPFormat[ HoldForm[f[x]] /. HoldPattern[x] -> bod, MathMLForm]
</msp:evaluate> =
```

```
<msp:evaluate> MSPFormat[ f[bod], MathMLForm] </msp:evaluate>.
</p>

<p>
Numerická hodnota
<msp:evaluate>
      MSPFormat[ HoldForm[f[x]] /. HoldPattern[x] -> bod, MathMLForm]
</msp:evaluate>
je <msp:evaluate> MSPFormat[ f[bod]//N, MathMLForm] </msp:evaluate>.
Po zaokrúhlení dostaneme
<msp:evaluate> MSPFormat[ N[f[bod],5], MathMLForm] </msp:evaluate>.
</p>

<p>
Deriváciiou funkcie je
<msp:evaluate> MSPFormat[ HoldForm[f'[x]], MathMLForm] </msp:evaluate> je
<msp:evaluate> MSPFormat[ f'[x], MathMLForm] </msp:evaluate> .
</p>

<p>
Výsledkom integrovania je:<br/>
<msp:evaluate>
      MSPFormat[ HoldForm[Integrate[f[x],x]], MathMLForm]
</msp:evaluate> =
<msp:evaluate>
      MSPFormat[ HoldForm[Integrate[f[x],x]]/. HoldPattern[f[x]] -> f[x],
      MathMLForm]
</msp:evaluate> =
<msp:evaluate> MSPFormat[ Integrate[f[x],x], MathMLForm]
<> MSPFormat[ HoldForm[+c],MathMLForm]</msp:evaluate>
</p>

</msp:allocateKernel>

</body>
</html>
```

Zdrojový kód stránky obsahuje rovnaké úvodné príkazy, ako boli uvedené v príkladoch v predchádzajúcej kapitole, preto sa tejto časti zdrojového kódu nebudeme podrobnejšie venovať. Potom nasleduje časť, ktorá je venovaná formuláru. V zdrojovom kóde je vyznačená tučne.

Jediná nová informácia sa týka **definície funkcie**. Všimnite si, že z prvého vstupné políčka si do výpočtového prostredia prinášame textovú premennú **`fun`**, ktorá má default hodnotu **`Sin[x]`**. Pretože ďalej chceme s touto premennou pracovať ako s funkciou, pri testovaní, či táto premenná nadobudla „nejakú“ hodnotu ju priamo definitoricky (okamžitým priradením) priradíme *MATHEMATICA* premennej **`f[x]`**. Obvyklým spôsobom otestujeme aj druhé vstupné políčko a jeho hodnotu priradíme premennej **`bod`**.

V úvodnej časti ešte stojí za povšimnutie organizácia zdrojového kódu. Na rozdiel od predchádzajúcich príkladov sme umiestnili do tagov ohraničujúcich začiatok a koniec formulára skutočne len príkazy týkajúce sa tohto formulára. Formulár sme od zvyšku stránky, v ktorej používame získané vstupy a zobrazujeme získané výstupy oddelili čiarou – tagom **`<hr/>`**. Tento spôsob organizácie stránky umožňuje prehľadnejšie oddelenie vstupov a výstupov a je ho výhodné používať pri vytváraní pedagogických aplikácií.

```
<p>
<msp:evaluate>
      "Definovaná bola funkcia " <> MSPFormat[f[x], MathMLForm]
</msp:evaluate>.
</p>
```

V predchádzajúcej časti zdrojového kódu je uvedený najjednoduchší spôsob, ako zverejniť výstup na webovú stránku. Môžeme kombinovať reťazec textu a výstup pomocou príkazu **`MSPFormat`**. Znak **`<>`** je štandardný spôsob spájania reťazcov v *MATHEMATICE*. V tomto odstavci spájame text (string) **`"Definovaná bola funkcia "`** so zobrazením funkcie, ktorú sme definovali v príslušnom formulárovom políčku. V nasledujúcej časti zdrojového kódu uvádzame aj ukážku, ako je možné v jednom príkaze **`<msp:evaluate>`** a **`</msp:evaluate>`** zobraziť aj komplikovanejší výstup, resp. niekoľko výstupov s rôznym charakterom.

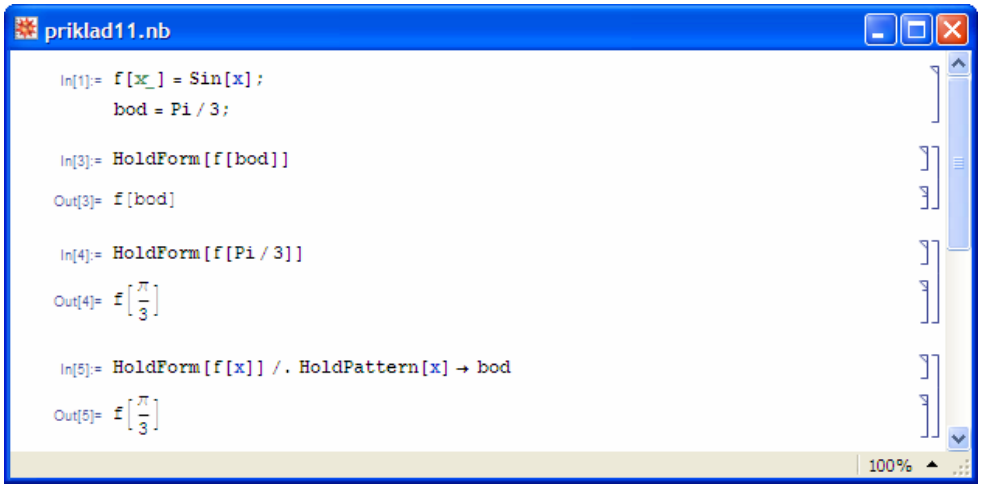
V nasledujúcom odstavci sa pokúsime vypočítať hodnotu funkcie v nejakom bode. Ak by sme potrebovali zobraziť skutočne len výsledok výpočtu – hodnotu funkcie $f(x)$ v zadanom bode – stačí použiť len priame dosadenie

```
<msp:evaluate> MSPFormat[ f[bod], MathMLForm] </msp:evaluate>
```

Pri príprave pedagogických materiálov – napríklad postupný výpočet pri dosadení hodnoty funkcie, musíme riešiť aj otázku, ako zobraziť na stránku vzťahy, ale nevypočítať ich hodnotu. Túto úlohu pomáha riešiť séria *MATHEMATICA* príkazov **`Hold`**, **`HoldAll`**, **`HoldForm`**, **`HoldPattern`**. Podrobnosti a ukážky použitia tejto série príkazov je možné nájsť v *MATHEMATICA* helpe.

Pozrime sa na zdrojový dokument a na použitie príkazov v *MATHEMATICE*.

```
<p>
Jej hodnota v bode
<msp:evaluate> MSPFormat[ bod, MathMLForm] </msp:evaluate> je
<msp:evaluate>
      MSPFormat[ HoldForm[f[x]] /. HoldPattern[x] -> bod, MathMLForm]
</msp:evaluate> =
<msp:evaluate>  MSPFormat[ f[bod], MathMLForm] </msp:evaluate>.
</p>
```



Príkaz **HoldForm** je pre naše účely najvhodnejší. Pozrime s na použitie tohto príkazu priamo v systéme *MATHEMATICA*. Tieto príkazy môžeme použiť aj vo vytvorenom jsp dokumente a dosiahnuť tak imitáciu postupného výpočtu.

Jej hodnota v bode $\frac{\pi}{3}$ je $f\left(\frac{\pi}{3}\right) = \frac{\sqrt{3}}{2}$.

Nasleduje výpočet numerickej hodnoty funkcie. Tento výpočet uvádzame predovšetkým preto, aby sme čitateľa upozornili, že web*MATHEMATICA* na rozdiel od *MATHEMATICE* uvádza pri požiadavke na zobrazenie numerickej hodnoty výsledok s presnosťou na 16 platných číslic. Ak nám stačí menej presný výsledok, musíme svoju požiadavku špecifikovať v príkaze **N[, počet platných číslic]**.

```
<p>
Numerická hodnota
<msp:evaluate>
      MSPFormat[ HoldForm[f[x]] /. HoldPattern[x] -> bod, MathMLForm]
</msp:evaluate>
je <msp:evaluate> MSPFormat[ f[bod]//N, MathMLForm] </msp:evaluate>.
Po zaokrúhlení dostaneme
<msp:evaluate> MSPFormat[ N[f[bod],5], MathMLForm] </msp:evaluate>.
</p>
```

Výsledok (pre default hodnoty) vyzerá takto:

Numerická hodnota $f\left(\frac{\pi}{3}\right)$ je 0.8660254037844386. Po zaokrúhlení dostaneme 0.86603.

Na prezentovanej jsp stránke ďalej vypočítame deriváciu funkcie a integrál. Používame len techniky vysvetlené v predchádzajúcich odstavcoch. Všimnite si, že nie je problém pomocou príkazu **HoldForm** zobraziť aj $f'(x)$ a $\int f(x)\,dx$. Príkaz **HoldPattern** použijeme, keď chceme získať medzivýsledok výpočtu, a v poslednom kroku integrujeme.

2. Zobrazenie viacerých obrázkov súčasne

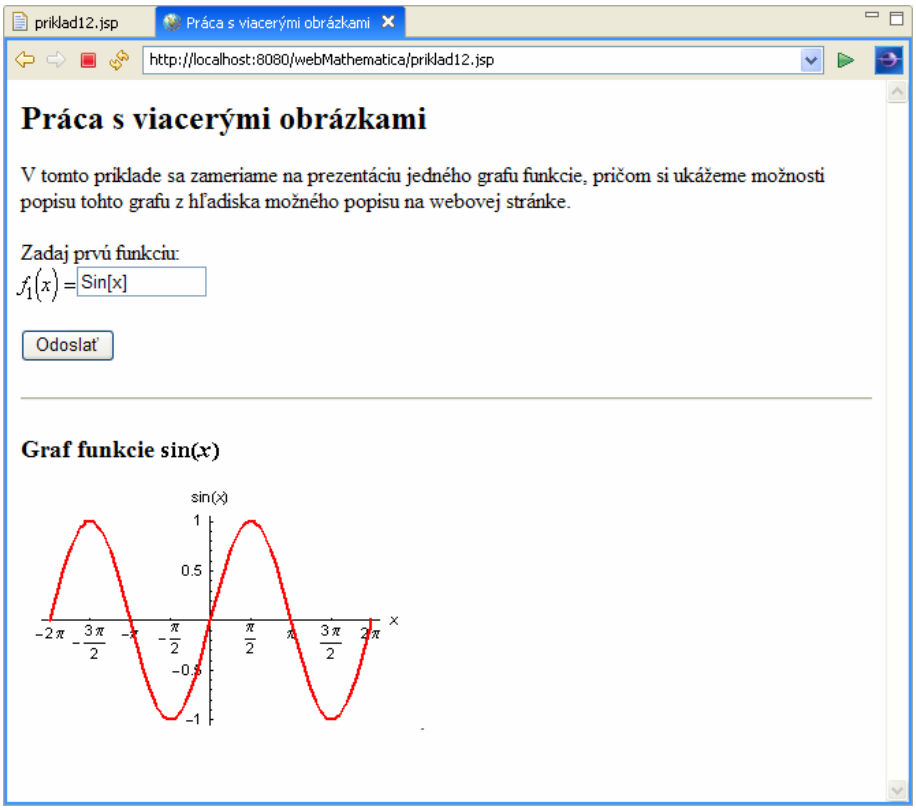
Pri príprave pedagogických materiálov sa veľmi často stretávame s úlohami, v ktorých je potrebné zobraziť do jedného obrázka niekoľko obrázkov súčasne, alebo s úlohou keď potrebujeme, napr. pre porovnávacie účely, zobraziť niekoľko obrázkov vedľa seba. Ak by sme chceli použiť len niekoľko za sebou nasledujúcich príkazov **MSPShow**, dosiahli by sme na výslednej webovej stránke zobrazenie niekoľkých obrázkov za sebou. Stránka by bola príliš dlhá a veľmi často neprehľadná.

Najskôr sa zameriame na popis obrázku, jednotlivých osí a podobne. **Z hľadiska efektívnosti a rýchlosti je najjednoduchšie všetky požiadavky na tvar, charakter a farbu výsledného grafu funkcie adresovať priamo výpočtovému jadru *MATHEMATICE*.** Vytvorený obrázok sa následne prenáša zo serveru do prehliadača užívateľa. Veľkosť obrázka závisí predovšetkým na jeho rozmeroch a je málo závislá od jeho obsahu. Čitateľom odporúčame, čo najviac príkazov pre popis a tvar obrázku zabezpečte pomocou príkazov a volieb *MATHEMATICE*.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl"
href="/webMathematica/Resources/XSL/mathml.xsl"?>

<%@ page language="java"  %>
<%@ page contentType="text/html;charset=utf-8" %>
<%@ taglib uri="/webMathematica-taglib" prefix="msp" %>

<html xmlns="http://www.w3.org/1999/xhtml">
      <head>
            <meta name="author" content="Monika Kovacova"/>
            <meta name="copyright" content="Monika Kovacova,
mathematica@mathematica.sk"/>
            <meta http-equiv="Content-type" content="text/html;
charset=utf-8" />
            <title>Práca s viacerými obrázkami</title>
      </head>

<body bgcolor="#ffffff">
<h2>Práca s viacerými obrázkami</h2>

<p>V tomto príklade sa zameriame na prezentáciu jedného grafu funkcie,
pričom si ukážeme možnosti popisu tohto grafu z hľadiska
možného popisu na webovej stránke.</p>
<msp:allocateKernel>
<msp:evaluate>
   MSPPageOptions[ "ContentType" -> "text/xml"];
</msp:evaluate>

<form action="priklad12.jsp" method="post">

<p>Zadaj prvú funkciu: <br/>
<math xmlns='http://www.w3.org/1998/Math/MathML'
    mathematica:form='StandardForm'
    xmlns:mathematica='http://www.wolfram.com/XML/'>
 <mrow>
  <mrow>
   <msub>
    <mi>f</mi>
    <mn>1</mn>
   </msub>
   <mo>(</mo>
   <mi>x</mi>
   <mo>)</mo>
  </mrow>
  <mo>=</mo>
 </mrow>
</math>
<input type="text" name="fun1" size="10"
            value="<msp:evaluate>MSPValue[$$fun1,"Sin[x]"]</msp:evaluate>"
/>
<msp:evaluate>
If[MSPValueQ[$$fun1],
            f1[x_] = ToExpression[$$fun1], f1[x_] = Sin[x]];
</msp:evaluate>
</p>

<input type="submit" name="tlacidlo" value="Odoslať" />
```

```
</form>
<hr/>

<h3> <msp:evaluate>
            "Graf funkcie "<> MSPFormat[ f1[x], MathMLForm]
      </msp:evaluate> </h3>

<p>
<msp:evaluate>
       MSPShow[
       Plot[f1[x], {x, -2Pi, 2Pi},
            AxesLabel -> TraditionalForm /@ {x, f1[x]},
            TextStyle -> {FontFamily -> "Arial", FontSize -> 11},
            Ticks -> {Table[i, {i, -2Pi, 2Pi, Pi/2}], Automatic},
            PlotStyle -> {Thickness[0.01], Hue[0]}]]
</msp:evaluate>.
<br/>
</p>

</msp:allocateKernel>

</body>

</html>
```

Po zobrazení stránky dostaneme tento výsledok.



Všimnite si, že priamo pomocou príkazu *MATHEMATICA* sme ovplyvnili veľkosť obrázku – **`200 x Automatic`** – ako počet bodov. Rovnako aj popis osí je vytvorený v prostredí *MATHEMATICE*. Všimnite si, že v oboch prípadoch môžeme definovať aj veľkosť aj typ

použitého písma. Pre popis obsahu vytvoreného obrázku sme v tomto príklade zvolili kombináciu HTML kódu a predpisu funkcie, pomocou príkazu **`MSPFormat`**.

Ak chceme zobraziť dva vytvorené obrázky do jedného spoločného obrázka – ekvivalent funkcie **`Show`** z *MATHEMATICE*, môžeme použiť nasledujúci kód.

```
<body bgcolor="#ffffff">
<h2>Práca s viacerými obrázkami</h2>

<p>V tomto príklade sa zameriame na prezentáciu grafov dvoch funkcií,
pričom si ukážeme možnosti zobrazenia výsledného grafu z hľadiska
webovej stránky.</p>
<msp:allocateKernel>
<msp:evaluate>
   MSPPageOptions[ "ContentType" -> "text/xml"];
</msp:evaluate>

<form action="priklad12.jsp" method="post">

<p>Zadaj prvú funkciu: <br/>
<math xmlns='http://www.w3.org/1998/Math/MathML'
    mathematica:form='StandardForm'
    xmlns:mathematica='http://www.wolfram.com/XML/'>
 <mrow>
  <mrow>
   <msub>
    <mi>f</mi>
    <mn>1</mn>
   </msub>
   <mo>(</mo>
   <mi>x</mi>
   <mo>)</mo>
  </mrow>
  <mo>=</mo>
 </mrow>
</math>
<input type="text" name="fun1" size="10"

      value="<msp:evaluate>MSPValue[$$fun1,"x*Sin[x]"]</msp:evaluate>" />
<msp:evaluate>
If[MSPValueQ[$$fun1],
           f1[x_] = ToExpression[$$fun1], f1[x_] = x*Sin[x]];
</msp:evaluate>
</p>

<p>Zadaj druhú funkciu: <br/>
<math xmlns='http://www.w3.org/1998/Math/MathML'
    mathematica:form='StandardForm'
    xmlns:mathematica='http://www.wolfram.com/XML/'>
 <mrow>
  <mrow>
   <msub>
    <mi>f</mi>
    <mn>2</mn>
   </msub>
   <mo>(</mo>
   <mi>x</mi>
   <mo>)</mo>
  </mrow>
```

```
  <mo>=</mo>
 </mrow>
</math>
<input type="text" name="fun2" size="10"

      value="<msp:evaluate>MSPValue[$$fun2,"ArcTan[x]"]</msp:evaluate>" />
<msp:evaluate>
If[MSPValueQ[$$fun2],
            f2[x_] = ToExpression[$$fun2], f2[x_] = ArcTan[x]];
</msp:evaluate>
</p>

<input type="submit" name="tlacidlo" value="Odoslať" />
</form>
<hr/>

<p>
<msp:evaluate>
obr1=Plot[f1[x], {x, -2Pi, 2Pi},
            TextStyle -> {FontFamily -> "Arial", FontSize -> 11},
            Ticks -> {Table[i, {i, -2Pi, 2Pi, Pi/2}], Automatic},
            PlotStyle -> {Thickness[0.01], Hue[0]}];

obr2=Plot[f2[x],{x, -2Pi, 2Pi},
            PlotStyle -> {Thickness[0.01], Hue[0.7]}];

MSPShow[ Show[obr1,obr2]]
</msp:evaluate>.
<br/>
</p>

</msp:allocateKernel>
</body>
```

V predchádzajúcej časti sme neuviedli zdrojový kód celej stránky, ale len jej časti ohraničenými tagmi **<body>** a **</body>**. Všimnite si, že v tomto zdrojovom kóde sme modifikovali formulár.

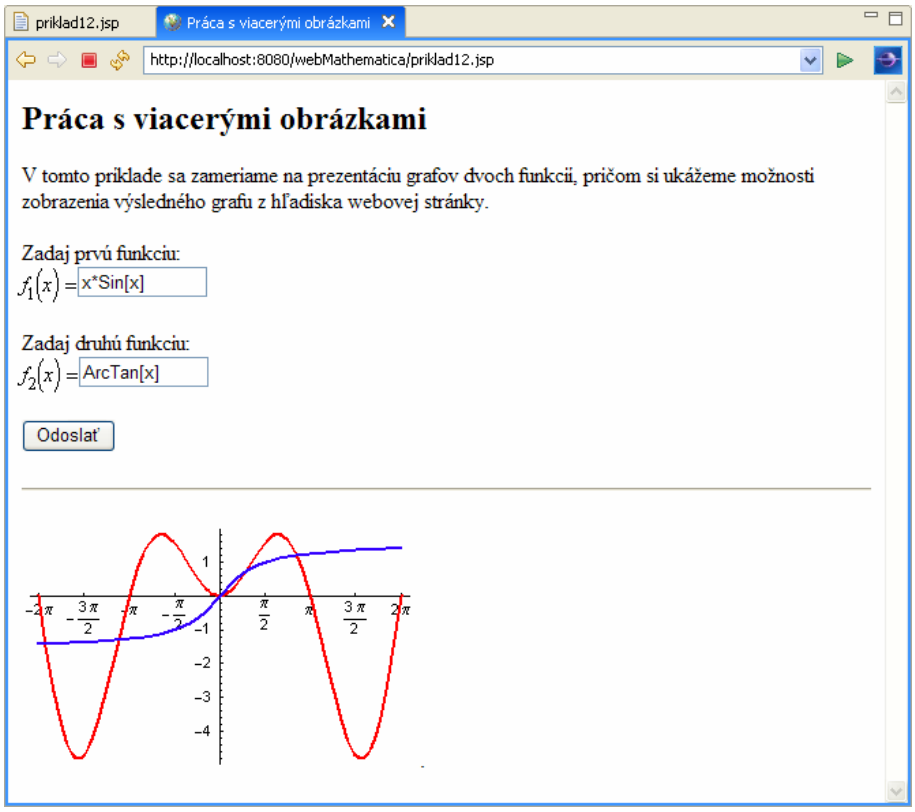
Vložili sme dve samostatné vstupné okná pre vstup dvoch samostatných funkcií. Jedna funkcia je označená ako **$$fun1** a jej ekvivalentom je funkcia **f1[x].** Druhá funkcia je označená ako **$$fun2** a jej ekvivalentom je funkcia **f2[x].** Ak chceme aby popis vstupného políčka bol dobre čitateľný, vložíme priamo do zdrojového kódu stránky MathML kód. Získame ho rovnako, ako v príklade 2 v predchádzajúcej kapitole.

V časti stránky, v ktorej údaje z formulára spracovávame, vytvoríme najskôr *MATHEMATICA* object - graf prvej funkcie. Tento object typu **-Graphics-** nazveme **obr1**. **Všimnite si, že na rozdiel od pracovného prostredia *MATHEMATICE* stačí na koniec príkazu dať bodkočiarku a vytvorený obrázok san a webovej stránke nezobrazí.** V prostredí *MATHEMATICE* musíme túto úlohu riešiť pomocou volieb **$DisplayFunction**. Rovnakým spôsobom vytvoríme druhý object typu **-Graphics-** a nazveme ho **obr2**. Aby bolo možné obrázky vo výslednom grafe rozlíšiť, už pri vytváraní objektov ich pomocou voliteľných príkzov v príkaze **Plot** upravíme tak, aby výsledný obrázok bol dostatočne čitateľný.

MATHEMATICA príkaz **`Show`** vytvorí object typu **`-Graphics-`**, ktorý je uložený na servery v dočasnom pracovnom adresári. Pomocou príkazu **`MSPShow`** ho následne zobrazíme na webovej stránke.

V ideálnom prípade by sme mali myslieť aj na správne ošetrenie prípustnosti zadávania predpisov funkcie, prípadne na možnosť zadávania intervalov kreslenia pre jednotlivé funkcie. Vzhľadom na skúsenosti, ktoré máme s tvorbou jsp stránok sa však týmto otázkam momentálne nebudeme venovať, nakoľko predstavujú rutinnú prácu.

Výsledok prezentovaného a vysvetleného zdrojové kódu vidíte na nasledujúcom obrázku.



Kombinovať môžeme nielen dva grafy vytvorené pomocou príkazu **`Plot`** a spojené do jedného objektu pomocou príkazu **`Show`**. Kombinovať predtým uvedeným spôsobom môžeme akékoľvek grafické objekty. V nasledujúcom príklade budeme prezentovať grafickú interpretáciu derivácie funkcie.

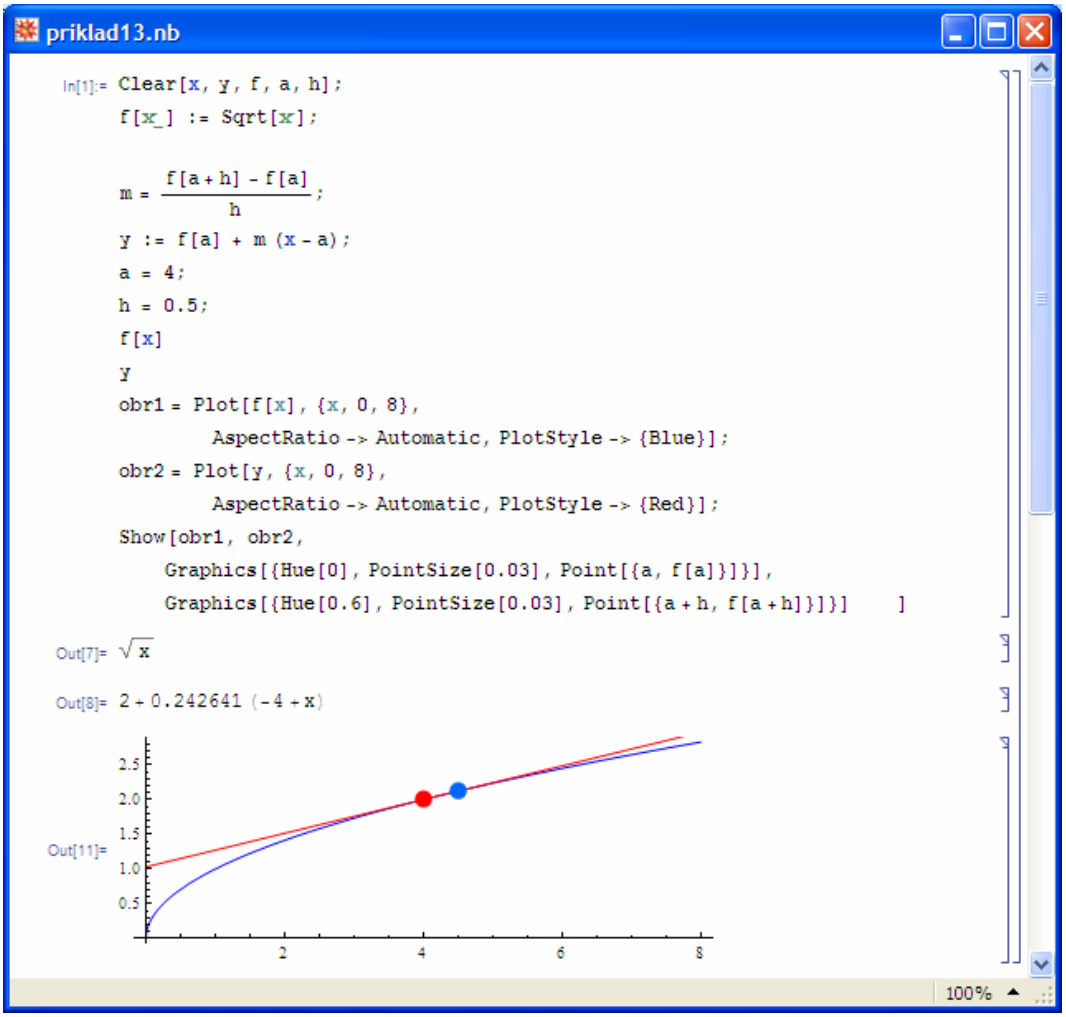
Grafická prezentácia viacerých grafických objektov rôznych typov

Veľmi často okrem prezentácie grafických objektov rovnakého typu potrebujeme vytvoriť stránku, na ktorej budeme prezentovať viaceré grafické objekty súčasne. Túto situáciu budeme prezentovať v nasledujúcom skripte a zameriame sa na zobrazenie dotyčnice ku grafu funkcie.

Len stručne zrekapitulujme hlavnú myšlienku. Chceme zostrojiť dotyčnicu ku grafu funkcie $f(x)$ v bode a. Potrebujeme mať zadefinovanú funkciu $f(x)$, a bod a, v ktorom chceme dotyčnicu zostrojiť. Aby sme mohli ukázať, ako sa mení/nemení dotyčnica vzhľadom na veľkosť kroku h, budeme konštruovať namiesto „skutočnej dotyčnice ku grafu funkcie“

priamku, ktorá prechádza bodmi $[a, f(a)]$ a $[a+h, f(a+h)]$. Využijeme tým interaktívnosť web*MATHEMATICA* formulára a umožníme študentovi otestovať situáciu pre rôzne hodnoty kroku h.

V *MATHEMATICE* by sme túto úlohu vyriešili nasledovne:



Ekvivalentný formulár vo web*MATHEMATICE* bude mať nasledujúci zdrojový kód. Samotné výpočtové jadro musíme obaliť obslužným formulárom tak, ako sme to vysvetlili v predchádzajúcej kapitole. Uvedieme len telo jsp stránky, pretože ostatné časti zdrojového kódu sa nemenia.

```
<body bgcolor="#ffffff">
<h2>Práca s viacerými obrázkami</h2>

<p>V tomto príklade budeme prezentovať možnosti zobrazenia viacerých
 - navzájom rôznych grafických objektov na jednej stránke.</p>
<msp:allocateKernel>
<msp:evaluate>
   MSPPageOptions[ "ContentType" -> "text/xml"];
</msp:evaluate>
```

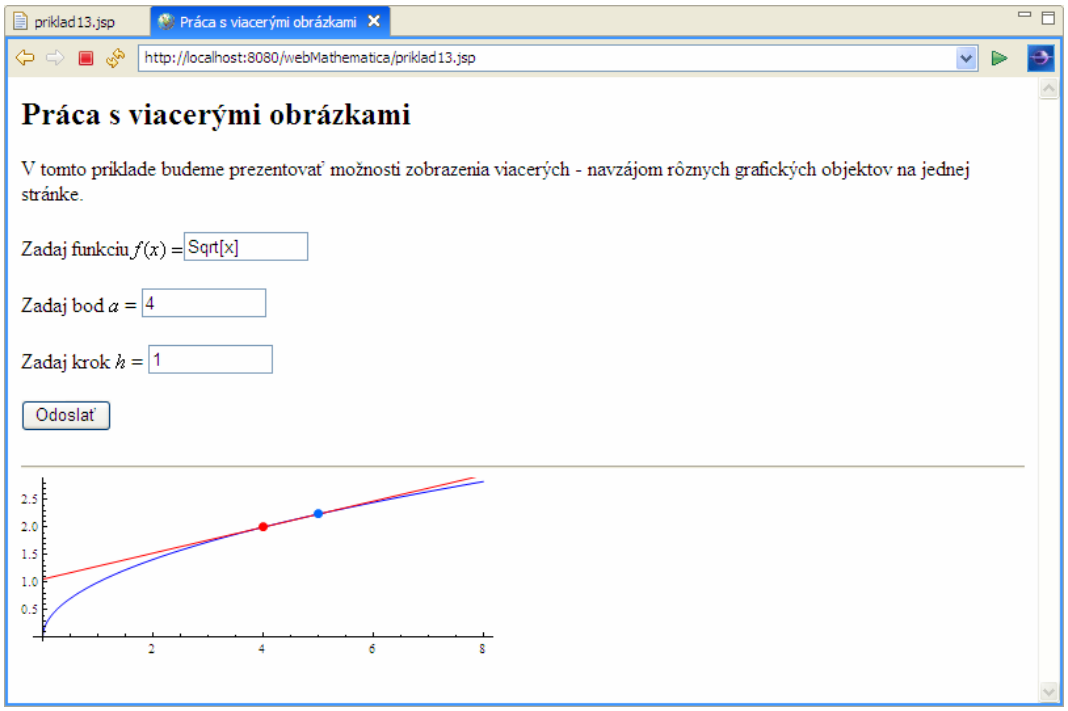
```
<form action="priklad13.jsp" method="post">

<p>Zadaj funkciu
<math xmlns='http://www.w3.org/1998/Math/MathML'
    mathematica:form='StandardForm'
    xmlns:mathematica='http://www.wolfram.com/XML/'>
 <mrow>
  <mrow>
    <mi>f</mi>
   <mo>(</mo>
   <mi>x</mi>
   <mo>)</mo>
  </mrow>
  <mo>=</mo>
 </mrow>
</math>
<input type="text" name="fun" size="10"
           value="<msp:evaluate>MSPValue[$$fun,"Sqrt[x]"]</msp:evaluate>"
/>
<msp:evaluate>
Clear[f];
If[MSPValueQ[$$fun],
           f[x_]=ToExpression[$$fun], f[x_] = Sqrt[x]];
</msp:evaluate>
</p>

<p>Zadaj bod <math xmlns='http://www.w3.org/1998/Math/MathML'
    mathematica:form='StandardForm'
    xmlns:mathematica='http://www.wolfram.com/XML/'>
 <mi>a</mi>
</math> =
<input type="text" name="bod" size="10"
           value="<msp:evaluate>MSPValue[$$bod,"4"]</msp:evaluate>" />
<msp:evaluate>
If[MSPValueQ[$$bod],
           a = ToExpression[$$bod], a = 4];
</msp:evaluate>
</p>

<p>Zadaj krok
<math xmlns='http://www.w3.org/1998/Math/MathML'
    mathematica:form='StandardForm'
    xmlns:mathematica='http://www.wolfram.com/XML/'>
 <mi>h</mi>
</math> =
<input type="text" name="krok" size="10"
           value="<msp:evaluate>MSPValue[$$krok,"1"]</msp:evaluate>" />
<msp:evaluate>
If[MSPValueQ[$$krok],
           h = ToExpression[$$krok], h = 1];
</msp:evaluate>
</p>
<input type="submit" name="tlacidlo" value="Odoslať" />
</form>
<hr/>

<msp:evaluate>

m =  (f[a + h] - f[a])/h;
y := f[a] + m (x - a);
obr1 = Plot[f[x], {x, 0,8 },
```

```
            AspectRatio -> Automatic, PlotStyle -> {Blue}];
obr2 = Plot[y, {x, 0, 8},
            AspectRatio -> Automatic, PlotStyle -> {Red}];
MSPShow[Show[obr1,obr2,
            Graphics[{Hue[0], PointSize[0.02], Point[{a, f[a]}]}],
            Graphics[{Hue[0.6], PointSize[0.02], Point[{a + h, f[a + h]}]}]
      ]]

</msp:evaluate>

</msp:allocateKernel>
</body>
```

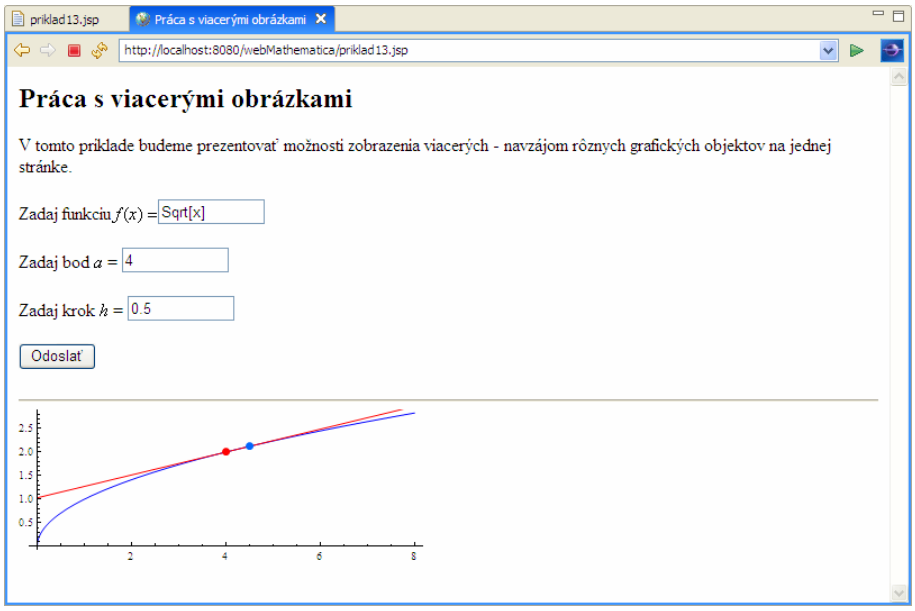
V predchádzajúcom zdrojovom kóde sme vytvorili a ošetrili vstupné formulárové políčko pre zadanie predpisu funkcie. Pridali sme formulárové políčko pre zadanie hodnoty bodu a a ďalšie formulárové políčko pre zadanie veľkosti kroku h.

Aj napriek tomu, že v reálnej aplikácii by bolo potrebné túto časť zdrojového kódu naprogramovať „poriadnejšie“, napríklad doplnením vysvetlenia a podporného textu vysvetľujúceho otázky čo počítame a prečo, v našej aplikácii sme túto časť vynechali. Z hľadiska cieľa príkladu – demonštrovať zobrazenie viacerých objektov na jednej stránke – je potrebné do príkladu zaradiť len minimálnu štruktúru programového kódu tak, aby sa v popise zdrojového kódu nestratil úmysel príkladu. Preto je aj naša zvolená aplikácia tak veľmi minimalistická.

Pri prvom spustení sa zobrazí nasledujúca stránka



Hodnoty premenných vstupujúcich prostredníctvom formulára môžeme zmeniť.



Vráťme sa teraz ku zdrojovému kódu našej stránky. V záverečnej časti kódu si všimnite, že môžeme definovať dva samostatné grafické objekty – **obr1** a **obr2**. Pretože oba príkazy Plot sme ukončili bodkočiarkou, výsledok - objekt typu – **Graphics** – sa nezobrazí na webovej stránke. Príkaz

```
Graphics[{Hue[0], PointSize[0.02], Point[{a, f[a]}]}]
```

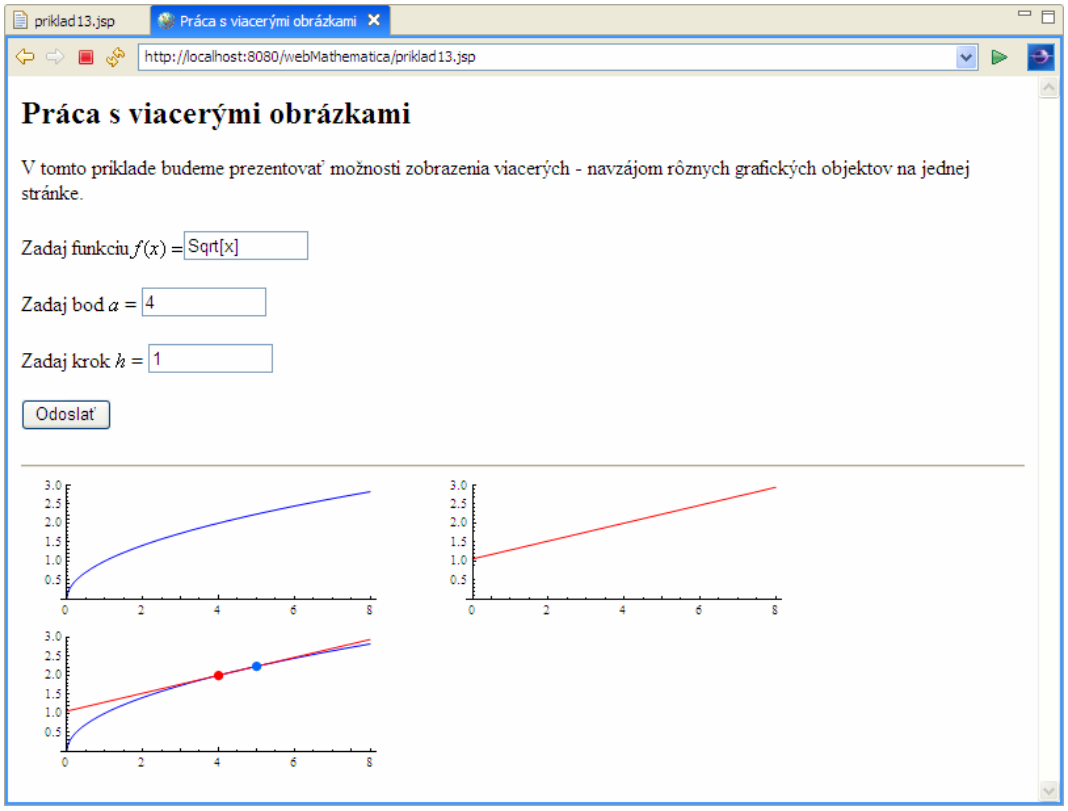
vytvorí grafický objekt typu – **Graphics** –, ktorý obsahuje červený bod $[a, f(a)]$. Podobným spôsobom vytvoríme aj druhý bod a pomocou príkazu **MSPShow** zobrazíme všetky grafické objekty do spoločného výsledného obrázku.

V zdrojovom kóde sme zvolili pevné hranice pre kreslenie obrázkov. V skutočnej reálnej aplikácii je potrebné dovoliť študentovi zvoliť hranice podľa vlastného uváženia. V tomto príklade sme pevné hranice zvolili z dôvodu minimalizácie dĺžky zdrojového kódu.

Ak potrebujeme zobraziť niekoľko obrázkov na stránke - napríklad dva obrázky vedľa seba je vhodné použiť združenie vytvorených grafických objektov pomocou príkazu **GraphicsArray[ ]**. Jeho argumentom musia byť grafické objekty, ktoré usporiadame podľa našich požiadaviek do štruktúry matice.

```
<msp:evaluate>
m =  (f[a + h] - f[a])/h;
y := f[a] + m (x - a);
obr1 = Plot[f[x], {x, 0,8 },PlotRange->{0,3},
            AspectRatio -> Automatic, PlotStyle -> {Blue}];
obr2 = Plot[y, {x, 0, 8},PlotRange->{0,3},
            AspectRatio -> Automatic, PlotStyle -> {Red}];
obr3 =Show[obr1,obr2,
            Graphics[{Hue[0], PointSize[0.03], Point[{a, f[a]}]}],
            Graphics[{Hue[0.6], PointSize[0.03], Point[{a + h, f[a + h]}]}]
      ];

MSPShow[GraphicsArray[{{obr1,obr2},{obr3}}, ImageSize->{600,Automatic}]]
</msp:evaluate>
```

Tu je výsledok upravenej stránky:



Pripomeňme ešte, že veľkosť výsledného obrázku môžeme ovplyvňovať aj pomocou príkazu **`ImageSize`** – všimnite si zdrojový kód.

3 Užívateľské nastavenie parametrov grafu

V tomto príklade si ukážeme niekoľko možností, ako je možné v zdrojovom kóde jsp súboru **povoliť užívateľské nastavenie**. Tieto možnosti budeme prezentovať na grafickej voľbe príkazu **`Plot`**. Dovolíme užívateľovi, aby si zvolil farbu grafu, ktorý chce nakresliť.

Príklad sme zvolili z niekoľkých dôvodov. Čitateľ sa v tomto príklade dozvie, ako je možné skombinovať príkaz voľby radio button z HTML jazyka s príkazmi web*MATHEMATICE*. Zaujímavá je aj ukážka, ako testovať, či zvolená premenná nadobudla konkrétnu hodnotu. V príklade si ukážeme aj použitie príkazu **`Which`**.

Najskôr uvedieme základnú verziu programu, potom sa budeme venovať jeho jednotlivým modifikáciám. Pozrime sa na zdrojový kód tohto súboru. Všetky dôležité časti sú vyznačené tučným typom písma. Uvedieme len telo jsp stránky, pretože ostatné časti zdrojového kódu sa nemenia.

```
<body bgcolor="#ffffff">
<h2>Užívateľské nastavenie parametrov grafu</h2>
```

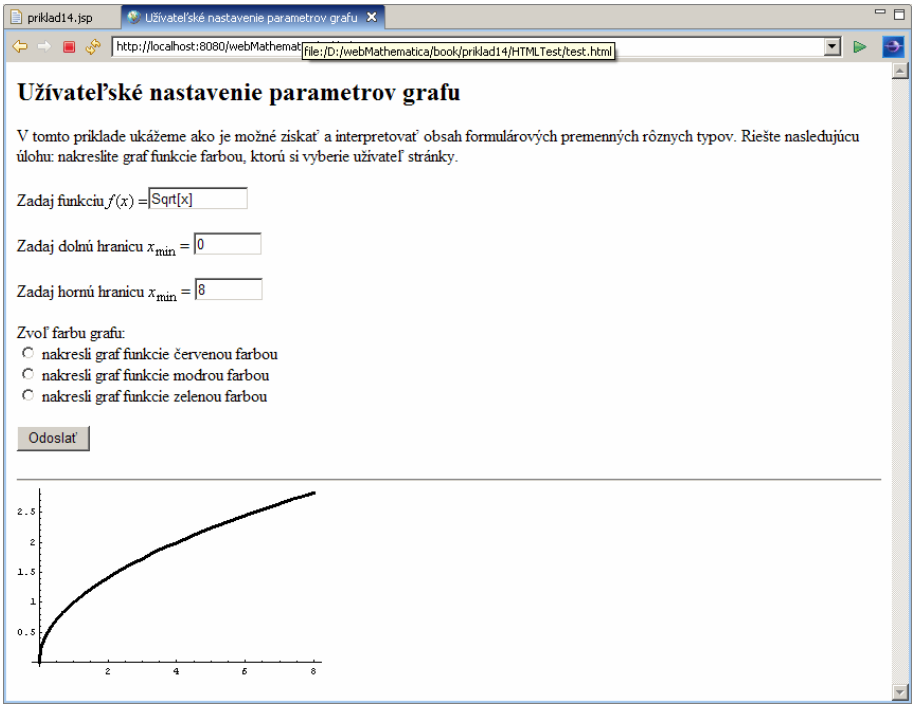
```
<p>V tomto príklade ukážeme ako je možné získať a interpretovať
obsah formulárových premenných rôznych typov. Riešte nasledujúcu úlohu:
nakreslite graf funkcie farbou, ktorú si vyberie užívateľ stránky.</p>

<msp:allocateKernel>
<msp:evaluate>
   MSPPageOptions[ "ContentType" -> "text/xml"];
</msp:evaluate>

<form action="priklad14.jsp" method="post">

<p>Zadaj funkciu
<math xmlns='http://www.w3.org/1998/Math/MathML'
    mathematica:form='StandardForm'
    xmlns:mathematica='http://www.wolfram.com/XML/'>
 <mrow>
  <mrow>
    <mi>f</mi>
   <mo>(</mo>
   <mi>x</mi>
   <mo>)</mo>
  </mrow>
  <mo>=</mo>
 </mrow>
</math>
<input type="text" name="fun" size="10"
           value="<msp:evaluate>MSPValue[$$fun,"Sqrt[x]"]</msp:evaluate>"
/>
<msp:evaluate>
Clear[f];
If[MSPValueQ[$$fun],
           f[x_]=ToExpression[$$fun], f[x_] = Sqrt[x]];
</msp:evaluate>
</p>

<p>
Zadaj dolnú hranicu
<math xmlns='http://www.w3.org/1998/Math/MathML'
    mathematica:form='StandardForm'
xmlns:mathematica='http://www.wolfram.com/XML/'>
 <msub>
  <mi>x</mi>
  <mi>min</mi>
 </msub>
</math> = <input type="text" name="xmin" size="5"
           value="<msp:evaluate>MSPValue[$$xmin,"0"]</msp:evaluate>" />
<msp:evaluate>
If[MSPValueQ[$$xmin], xmin = ToExpression[$$xmin], xmin = 0];
</msp:evaluate>
</p>

<p>
Zadaj hornú hranicu
<math xmlns='http://www.w3.org/1998/Math/MathML'
    mathematica:form='StandardForm'
xmlns:mathematica='http://www.wolfram.com/XML/'>
 <msub>
  <mi>x</mi>
  <mi>min</mi>
 </msub>
</math> = <input type="text" name="xmax" size="5"
```

```
            value="<msp:evaluate>MSPValue[$$xmax,"8"]</msp:evaluate>" />
<msp:evaluate>
If[MSPValueQ[$$xmax], xmax = ToExpression[$$xmax], xmax = 8];
</msp:evaluate>
</p>

<p>
Zvoľ farbu grafu: <br/>
<input type="radio" name="color" value="Red"
      <msp:evaluate>If[$$color==="Red","checked=\"checked\""]</msp:evaluate
> />
      nakresli graf funkcie červenou farbou<br/>
<input type="radio" name="color" value="Blue"
      <msp:evaluate>If[$$color==="Blue","checked=\"checked\""]</msp:evaluat
e> />
      nakresli graf funkcie modrou farbou <br/>
<input type="radio" name="color" value="Green"
      <msp:evaluate>If[$$color==="Green","checked=\"checked\""]</msp:evalua
te> />
      nakresli graf funkcie zelenou farbou
</p>

<input type="submit" name="tlacidlo" value="Odoslať" />
</form>
<hr/>

<msp:evaluate>

Which[
      $$color==="Red", MSPShow[Plot[f[x],{x,xmin,xmax},
                        PlotStyle->{Red, Thickness[0.012]}]],
      $$color==="Blue", MSPShow[Plot[f[x],{x,xmin,xmax},
                        PlotStyle->{Blue, Thickness[0.012]}]],
      $$color==="Green", MSPShow[Plot[f[x],{x,xmin,xmax},
                        PlotStyle->{Green, Thickness[0.012]}]],
      True, MSPShow[Plot[f[x],{x,xmin,xmax},
                        PlotStyle->{Black, Thickness[0.012]}]]
      ]
</msp:evaluate>

</msp:allocateKernel>

</body>
```

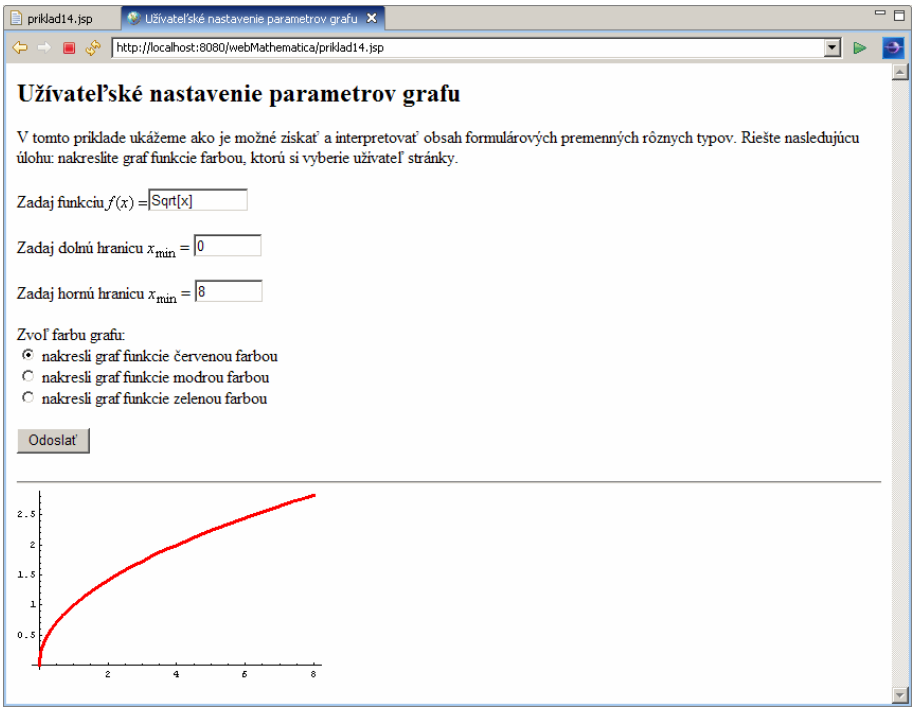
Najskôr vysvetlime ako stránka funguje. Pri prvom zobrazení stránky sa do premenných **xmin** a **xmax** uložia hodnoty dolnej a hornej hranice intervalu, na ktorom budeme kresliť funkciu **f[x]** zadanú prostredníctvom prvého vstupného formulárového políčka. Všetky tieto činnosti sme už niekoľko krát v rôznych príkladoch použili a preto sa im nebudeme podrobne venovať.

Vo vstupnom políčku typu **radio-button** dovolíme používateľovi stránky, aby si zvolil farbu grafu, ktorá bude následne použitá v príkaze **Plot** pre vykreslenie grafu zadanej funkcie. Toto zaklikávacie políčko má svoj názov – **color** a môže v našom príklade nadobúdať farby **Red**, **Blue** a **Green**.

Pri prvom zobrazení stránky premenná **$$color** ešte nenadobudla žiadnu hodnotu, preto sa graf zobrazí čiernou farbou. Realizuje sa tak default vetva príkazu **Which**. Výsledok prvého zobrazenia je uvedený na nasledujúcom obrázku.



Užívateľ po prvom zobrazení stránky môže kliknúť na jedno zo zaškrtávacích políčok a zvoliť farbu výsledného zobrazeného grafu funkciu. Po označení jednej z možností premenná **$$color** nadobudne jednu zo zvolených hodnôt (**Red**, **Blue** alebo **Green**) a následne sa graf funkcie vykreslí pomocou zvolenej farby.



Možnosť výberu uvedená na predchádzajúcom obrázku má nasledujúci zdrojový kód.

```
<input type="radio" name="color" value="Red"
      <msp:evaluate>
            If[$$color===\"Red\",\"checked=\"checked\"\"]
      </msp:evaluate> />
```

Všimnite si, že vo všetkých troch prípadoch voľby farby je tento **radiobutton pomenovaný rovnakým názvom.** Je to potrebné preto, lebo len v tomto prípade bude správne fungovať alternácia voľby jednotlivých farieb a premenná **color** nadobudne len jednu hodnotu – hodnotu posledného zaklikávacieho políčka, na ktoré užívateľ klikol.

Časť príkazu uvedená v ohraničujúcich tagoch **<msp:evaluate>** a **</msp:evaluate>** zabezpečuje, aby si stránka zapamätala užívateľovu poslednú voľbu farby grafu. V tejto časti testujeme, či web*MATHEMATICA* premenná **$$color** nadobudla hodnotu **Red**. Ak áno, označíme príslušné políčko ako zaškrtnuté.

Všimnite si, že úvodzovky ohraničujúce vyhradené slovo **checked** musia byť zapísané v plnom tvare, teda **=\"checked\".** Sú to vnorené úvodzovky, preto ich je potrebné takýmto spôsobom oddeľovať.

Na vykreslenie grafu funkcie použijeme príkaz **Which**, ktorý umožňuje niekoľkonásobné vetvenie a môže mať definovanú aj tzv. default vetvu. Táto vetva sa uvádza ako posledná a jej realizácia nastane, ak žiadna z predchádzajúcich podmienok nebola splnená. Syntaktickej stránke tohto príkazu sme sa venovali v kapitole o programovaní v *MATHEMATICE*.

Všimnite si tiež, že pri testovaní hodnoty premennej **$$color** musíme používať **===.** Musíme porovnávať aj hodnotu aj typ premennej.

Modifikácia 1 – zmena označenia farby

V nasledujúcej časti si ukážeme, ako je možné predchádzajúci príklad modifikovať. V príklade vôbec nie je dôležité označenie farby slovom **Red**, **Blue** alebo **Green**. Farbu môžeme definovať prostredníctvom ktoréhokoľvek z príkazov pre voľbu farby – **Hue[ ], RGBColor[ ], CMYKColor[ ]**.

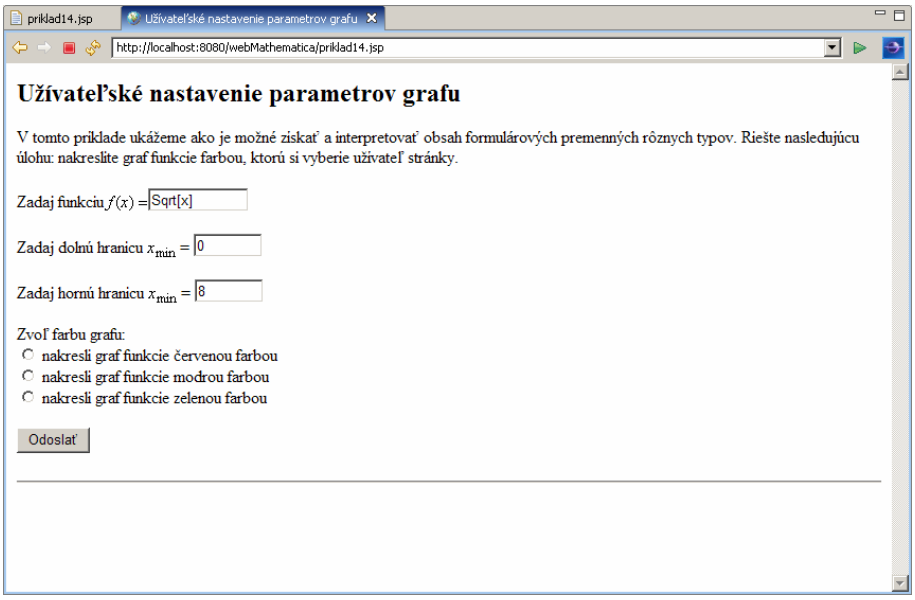
Zdrojový kód v prípade potreby použitia takéhoto farebného označenia zmeníme nasledovne:

```
<p>
Zvoľ farbu grafu: <br/>
<input type="radio" name="color" value="Hue[0]"
      <msp:evaluate>If[$$color===\"Hue[0]\",\"checked=\"checked\"\"]</msp:evalu
ate> />
      nakresli graf funkcie červenou farbou<br/>
<input type="radio" name="color" value="Hue[0.6]"
      <msp:evaluate>If[$$color===\"Hue[0.6]\",\"checked=\"checked\"\"]</msp:eva
luate> />
      nakresli graf funkcie modrou farbou <br/>
```

```
<input type="radio" name="color" value="Hue[0.3]"
      <msp:evaluate>If[$$color==="Hue[0.3]","checked=\"checked\""]</msp:eva
luate> />
      nakresli graf funkcie zelenou farbou
</p>

<input type="submit" name="tlacidlo" value="Odoslať" />
</form>
<hr/>

<msp:evaluate>

Which[
      $$color==="Hue[0]", MSPShow[Plot[f[x],{x,xmin,xmax},
                        PlotStyle->{Hue[0], Thickness[0.012]}]],
      $$color==="Hue[0.6]", MSPShow[Plot[f[x],{x,xmin,xmax},
                        PlotStyle->{Hue[0.6], Thickness[0.012]}]],
      $$color==="Hue[0.3]", MSPShow[Plot[f[x],{x,xmin,xmax},
                        PlotStyle->{Hue[0.3], Thickness[0.012]}]],
      True, MSPShow[Plot[f[x],{x,xmin,xmax},
                        PlotStyle->{Black, Thickness[0.012]}]]
      ]
</msp:evaluate>
```

Modifikácia 2 – odstránenie default vetvy grafu

Ak z príkazu **`Which`** odstránime default vetvu, pri prvom zobrazení stránky nastane nasledujúca situácia. Premenná **`$$color`** nenadobudne žiadnu hodnotu, preto ani jedna z podmienok v príkaze **`Which`** nie je splnená. Nie je možné realizovať vykreslenie žiadnej z troch možností a stránka sa zobrazí v nasledujúcom tvare.



Zdrojový kód upraveného príkazu **`Which`** má nasledovný tvar:

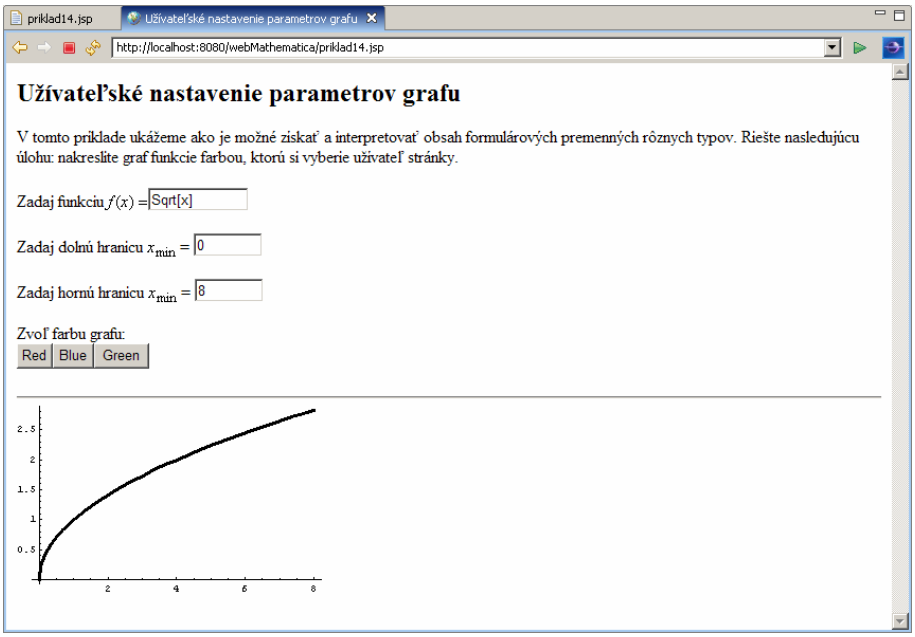
```
<msp:evaluate>

Which[
      $$color==="Hue[0]", MSPShow[Plot[f[x],{x,xmin,xmax},
                          PlotStyle->{Hue[0], Thickness[0.012]}]],
      $$color==="Hue[0.6]", MSPShow[Plot[f[x],{x,xmin,xmax},
                          PlotStyle->{Hue[0.6], Thickness[0.012]}]],
      $$color==="Hue[0.3]", MSPShow[Plot[f[x],{x,xmin,xmax},
                          PlotStyle->{Hue[0.3], Thickness[0.012]}]]
      ]
</msp:evaluate>
```

Modifikácia 3 – užívateľská voľba pomocou tlačítok typu „submit"

Táto modifikácia vykazuje azda najvýznamnejšie rozdiely oproti predchádzajúcim príkladom. V formulári môžeme užívateľovi povoliť voľbu užívateľského nastavenia aj pomocou tlačítok typu **`submit`**. Tieto tlačítka sa obvykle používajú len na odoslanie formulára na spracovanie. Ich stlačenie spôsobí, že stránka je znovu zobrazená s poslednými zvolenými parametrami. Ak ako názov tlačítka zvolíme premennú **`color`**, stlačenie príslušného tlačítka spôsobí, že premenná **`$$color`** nadobudne hodnotu toho tlačítka, ktoré bolo aktuálne stlačené.

Tento postup nie je príliš častý, uvádzame ho tu len kvôli kompletnosti príkladu, nakoľko sa môžu vyskytnúť aplikácie, v ktorých bude práve tento typ užívateľskej voľby potrebný.

Pri prvom zobrazení modifikovanej stránky uvidíme:

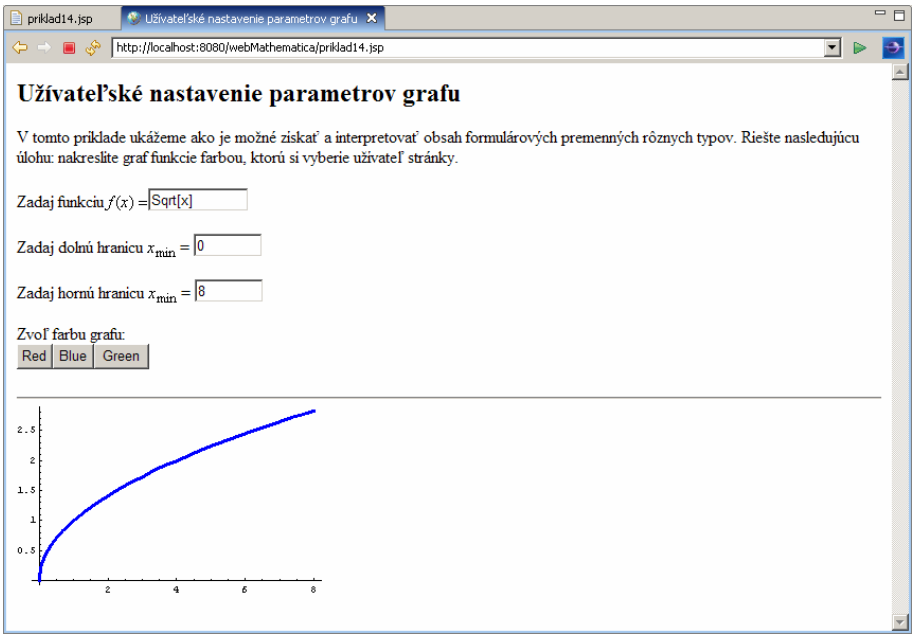


Tvar zobrazených tlačítok voľby je závislý od nastavení prehliadača. Môžeme ho ovplyvniť aj pomocou definovania CSS štýlu. Zdrojový kód zobrazenej stránky má tvar (uvádzame len modifikovanú časť zdrojového kódu):

```
<p>
Zvoľ farbu grafu:<br/>
<input type="submit" name="color" value="Red"/>
<input type="submit" name="color" value="Blue"/>
<input type="submit" name="color" value="Green"/>
</p>

</form>
<hr/>

<msp:evaluate>

Which[
      $$color==="Red", MSPShow[Plot[f[x],{x,xmin,xmax},
                        PlotStyle->{Red, Thickness[0.012]}]],
      $$color==="Blue", MSPShow[Plot[f[x],{x,xmin,xmax},
                        PlotStyle->{Blue, Thickness[0.012]}]],
      $$color==="Green", MSPShow[Plot[f[x],{x,xmin,xmax},
                        PlotStyle->{Green, Thickness[0.012]}]],
      True, MSPShow[Plot[f[x],{x,xmin,xmax},
                        PlotStyle->{Black, Thickness[0.012]}]]
      ]

</msp:evaluate>
```

Všimnite si v zdrojovom kóde, že pre voľbu farby používame tlačítko typu **`submit`**. Hodnota tohto tlačítka, definovaná v možnosti **`value`** (v našom prípade **`Red`**, **`Blue`** alebo **`Green`**) **je zároveň textom, ktorý sa zobrazí na tlačítku.**

Po kliknutí na tlačítko s názvom **`Blue`**, premenná **`$$color`** nadobudne hodnotu **`Blue`** a stránka sa zobrazí v tvare:



Upozorňujeme čitateľa, že nie je možné používať ako hodnotu tlačítka texty s diakritikou napríklad **`value="Červená"`**, pretože obsah tejto voľby je pri spracovaní priradený premenej **`$$color`** – ale už v spracovanej forme-**`"\[CapitalCHacek]erven\[AAcute]"`**. Porovnávať

obsah premennej zapísanej v tomto tvare je problematický. Preto ak budete vytvárať slovenské stránky, pamätajte na toto obmedzenie a názvy tlačítok voľte tak, aby mali dostatočnú vypovedaciu hodnotu a zároveň ste v nich nepoužili znaky kódových sád mimo Latin1.

4 Použitie *MATHEMATICA* packages

Modifikácia 1 – výpočtový balík

Veľmi často je potrebné použiť v rámci jsp stránok aj príkazy, ktoré nie sú súčasťou základného balíka príkazov programu *MATHEMATICA*. V nasledujúcom príklade si ukážeme použitie balíka na riešenie nerovníc.

Ak by sme chceli realizovať výpočet priamo pomocou systému *MATHEMATICA*, použijeme nasledujúce príkazy:

```
In[1]:= << Algebra`InequalitySolve`

In[2]:= InequalitySolve[x (x^2 - 2) (x^2 - 3) > 0, x]

Out[2]= -√3 < x < -√2 || 0 < x < √2 || x > √3
```

Pri vytváraní ekvivalentnej jsp stránky, nemôžeme použiť skrátenú formu zápisu načítania balíka na riešenie nerovníc. Musíme použiť plný tvar tohoto príkazu

Needs["Algebra`InequalitySolve`"]
Zdrojový kód takej stránky bude nasledujúci. Rovnako ako v ostatných príkladoch tejto kapitoly uvádzame len základné telo jsp stránky, nakoľko úvodná časť sa vôbec nemení.

```
<body bgcolor="#ffffff">
<h2>Použitie <i>MATHEMATICA</i> packages</h2>

<p>V tomto príklade ukážeme ako je možné použiť aj špeciálne balíky systému
<i>MATHEMATICA</i>,
t.j. balíky, ktorých použite treba samostatne inicializovať.</p>
<msp:allocateKernel>

<msp:evaluate>
   MSPPageOptions[ "ContentType" -> "text/xml"];
   Needs["Algebra`InequalitySolve`"]
</msp:evaluate>

<form action="priklad15.jsp" method="post">

<p>Zadaj nerovnicu, ktorú chceš vypočítať: <br/>

<input type="text" name="nerovnica" size="40"
      value="<msp:evaluate>
      MSPValue[$$nerovnica,"x (x^2 - 2) (x^2 - 3) > 0"]</msp:evaluate>" />
            <msp:evaluate>
If[MSPValueQ[$$nerovnica],
            nerovnica=ToExpression[$$nerovnica],
```

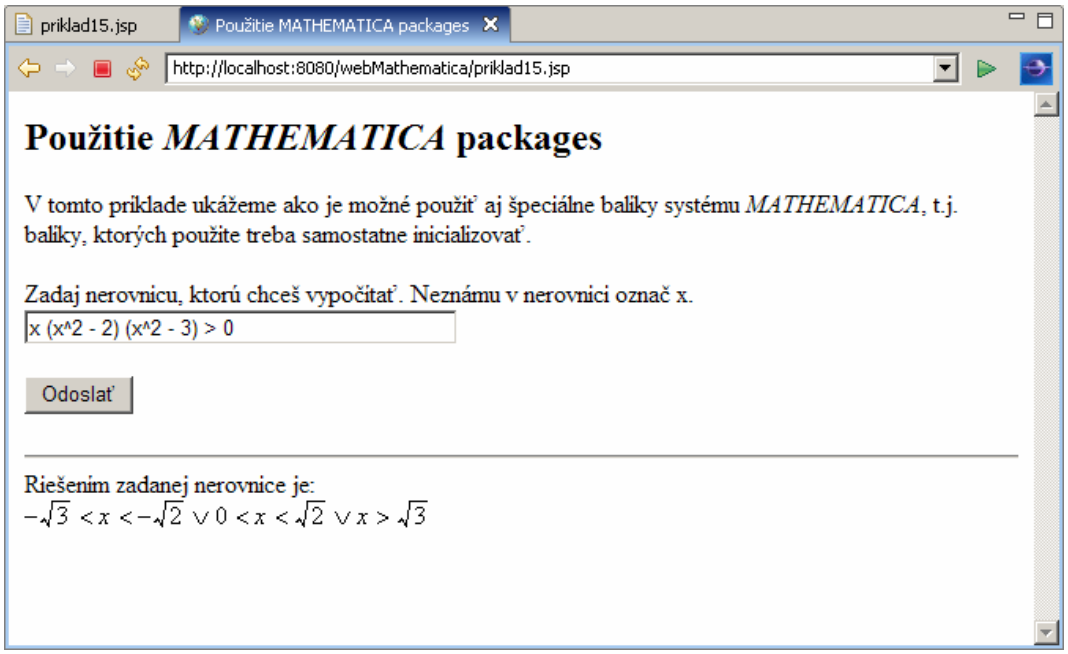
```
            nerovnica = ToExpression["x (x^2 - 2) (x^2 - 3) > 0"] ];
</msp:evaluate>
</p>

<input type="submit" name="tlacidlo" value="Odoslať" />
</form>
<hr/>

Riešením zadanej nerovnice je: <br/>
<msp:evaluate>
      MSPFormat[InequalitySolve[nerovnica, x], MathMLForm]
</msp:evaluate>

</msp:allocateKernel>

</body>
```

Všetky dôležité časti zdrojového kódu sú vyznačené tučným typom písma.

Všimnite si, že hneď v úvode stránky sme v rámci výpočtového prostredia **<msp:evaluate> </msp:evaluate>** použili plný tvar príkazu **Needs** určený na načítanie balíka systému *MATHEMATICA*. Načítali sme balík **Algebra`InequalitySolve`"** obsahujúci príkazy na riešenie nerovníc.

Nasledujúca časť zdrojového kódu má pomerne jednoduchý tvar. Najskôr obvyklým spôsobom vytvoríme formulárové políčko pre zadanie **nerovnice**. Z HTML premenenej **$$nerovnica** vytvoríme *MATHEMATICA* premennú **nerovnica** a vyriešime ju pomocou príkazu **InequalitySolve [nerovnica, x]**.

Pri zadávaní nerovnice automaticky predpokladáme u užívateľa znalosť synataxe systému *MATHEMATICA* pri zadávaní nerovníc a použitú neznámu v nerovnici defaultne označíme x. Výsledok zobrazíme obvyklým spôsobom pomocou príkazu **MSPFormat[ ]**

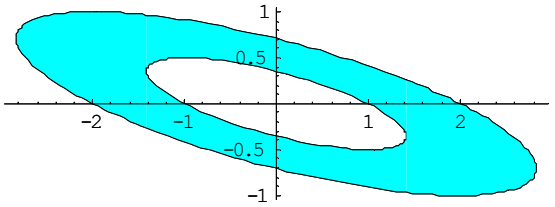


Modifikácia 2 – grafický balík

Táto modifikácia je veľmi podobná predchádzajúcemu príkladu. Ukážeme si použite grafického balíka na kreslenie funkcií zadaných ohraničujúcimi nerovnicami. Všetky zásady, ktoré sme uviedli pri prvej modifikácii platia aj pri použití grafických balíkov.

Výpočet priamo pomocou systému MATHEMATICA zrealizujeme príkazmi:

```
In[3]:= << Graphics`InequalityGraphics`

In[4]:= InequalityPlot[ 1 ≤ (x+ 2 y)^2 + 4 y^2 ≤ 4, {x, -3, 3}, {y, -3, 3} ]
```



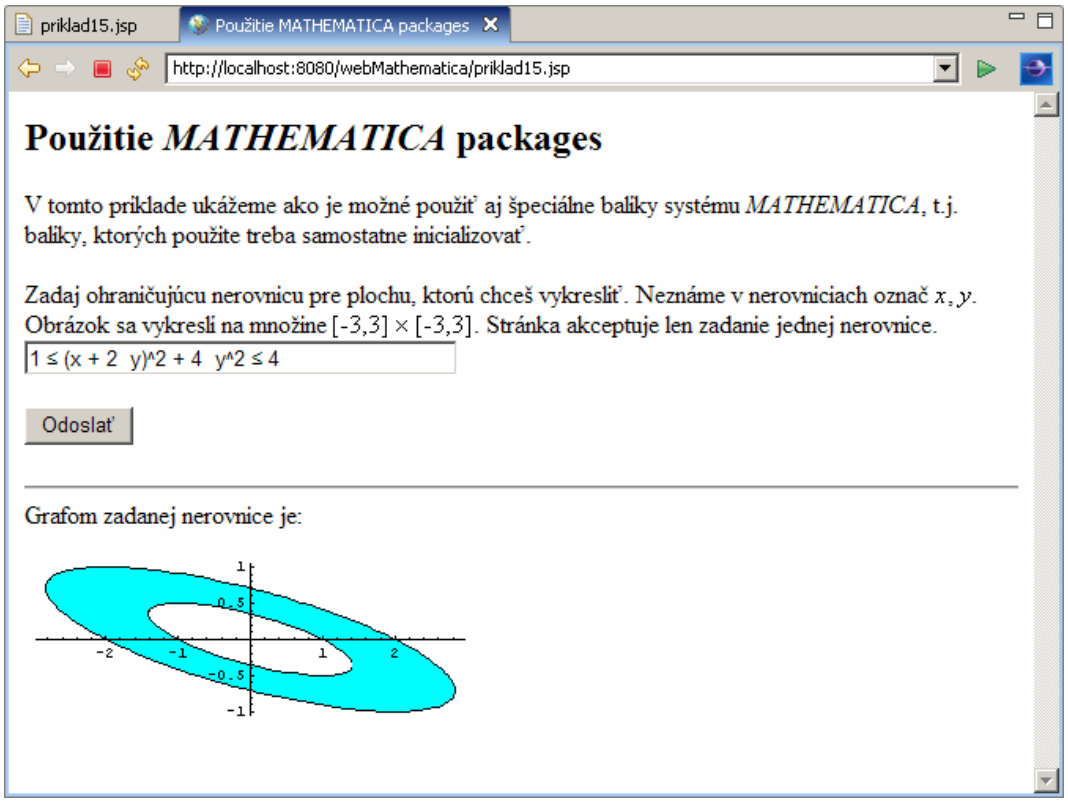

```
Out[4]= - Graphics -
```

Zdrojový kód jsp stránky, ktorá bude realizovať rovnaký výpočet bude mať tvar.

```
<body bgcolor="#ffffff">
<h2>Použitie <i>MATHEMATICA</i> packages</h2>

<p>V tomto príklade ukážeme ako je možné použiť aj špeciálne balíky systému
<i>MATHEMATICA</i>,
t.j. balíky, ktorých použite treba samostatne inicializovať.</p>
<msp:allocateKernel>
<msp:evaluate>
   MSPPageOptions[ "ContentType" -> "text/xml"];
   Needs["Graphics`InequalityGraphics`"]
</msp:evaluate>


<form action="priklad15.jsp" method="post">

<p>Zadaj ohraničujúcu nerovnicu pre plochu, ktorú chceš vykresliť.
Neznáme v nerovniciach označ
<math xmlns='http://www.w3.org/1998/Math/MathML'
    mathematica:form='StandardForm'
    xmlns:mathematica='http://www.wolfram.com/XML/'>
 <mi>x</mi>
</math>,
<math xmlns='http://www.w3.org/1998/Math/MathML'
    mathematica:form='StandardForm'
    xmlns:mathematica='http://www.wolfram.com/XML/'>
 <mi>y</mi>
</math>. Obrázok sa vykreslí na  množine
<math xmlns='http://www.w3.org/1998/Math/MathML'
    mathematica:form='StandardForm'
    xmlns:mathematica='http://www.wolfram.com/XML/'>
 <mtext>[-3,3] &#215; [-3,3]</mtext>
</math>. Stránka akceptuje len zadanie jednej nerovnice.<br/>
```

```
<input type="text" name="nerovnica" size="40"
            value="<msp:evaluate>
                  MSPValue[$$nerovnica,"1 ≤ (x + 2  y)^2 + 4  y^2 ≤ 4"]
                     </msp:evaluate>" />
<msp:evaluate>
If[MSPValueQ[$$nerovnica],
            nerovnica=ToExpression[$$nerovnica],
            nerovnica = ToExpression["1 ≤ (x + 2  y)^2 + 4  y^2 ≤ 4"] ];
</msp:evaluate>
</p>

<input type="submit" name="tlacidlo" value="Odoslať" />
</form>
<hr/>

Grafom zadanej nerovnice je: <br/><br/>
<msp:evaluate>
      MSPShow[InequalityPlot[nerovnica, {x, -3, 3}, {y, -3, 3} ]]
</msp:evaluate>

</msp:allocateKernel>

</body>
```

Po zobrazení stránky dostaneme výsledok veľmi podobný výsledku výpočtu v systéme *MATHEMATICA*.



Všetky dôležité časti zdrojového kódu sú vyznačené tučným typom písma. V úvodnej časti stránky pomocou plného tvaru príkazu **Needs** načítame grafický balík, ktorý obsahuje príkazy potrebné na vykreslenie nerovníc.

```
Needs["Graphics`InequalityGraphics`"]
```

Pretože výsledkom výpočtu je obrázok, na jeho zobrazenie použijeme príkaz **MSPShow**.

```
MSPShow[InequalityPlot[nerovnica, {x, -3, 3}, {y, -3, 3} ]]
```

Cieľom príkladu bolo ukázať použitie grafického balíka, preto celý zostávajúci kód je značne minimalistický a nepovoľuje užívateľovi ani zvoliť si tvar zadávanej nerovnice, ani premenné, ani interval na ktorom budeme nerovnicu vykresľovať. Pri vytváraní reálnej aplikácie pre študentov je potrebné všetky tieto úlohy v zdrojovom kóde ošetriť a ponechať užívateľovi možnosť voľby.

Čitateľa by sme v súvislosti s používaním balíkov (packages) systému *MATHEMATICA* chceli upozorniť, **že funkčnosť použitého balíka je závislá na inštalácii systému *MATHEMATICA*, ktorá bola nainštalovaná na serveri realizujúcom výpočet**. Ak potrebujete na stránkach používať vlastné vytvorené balíky je potrebné zaistiť ich umiestnenie na serveri do adresárovej štruktúry inštalácie systému *MATHEMATICA*. Nestačí umiestniť príslušný balík do štruktúry vášho adresára, z ktorého sa príslušná jsp stránka spúšťa. Ak používate server Strojníckej fakulty je potrebné obvyklým spôsobom kontaktovať administrátora serveru, ktorý zabezpečí umiestnenie vášho balíka na server.

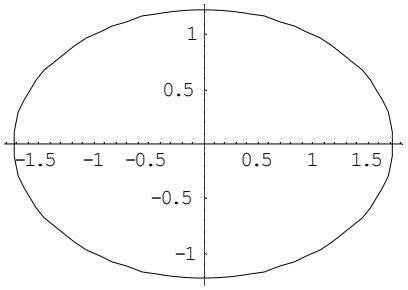
Modifikácia 3 – kombinácia grafických balíkov

V tejto modifikácii základnej myšlienky o použití balíkov vo výpočtoch na jsp stránkach si ukážeme, že môžeme ľubovoľne modifikovať objekty, ktoré vzniknú ako výsledok výpočtu realizovaného pomocou grafických balíkov a štandardného príkazu na kreslene **Plot**.

Často potrebujeme do jedného obrázka zobraziť graf funkcie zadanej implicitne a graf funkcie zadanej predpisom $y = f(x)$. Jednoduchú verziu tohto príkladu si ukážeme v tejto modifikácii.

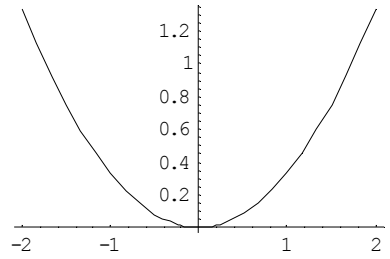
Ak by sme výpočet realizovali pomocou programového systému *MATHEMATICA*, je potrebné uskutočniť nasledujúce kroky.

```
In[1]:= << Graphics`ImplicitPlot`

In[2]:= obr1 = ImplicitPlot[x^2 + 2 y^2 == 3, {x, -2, 2}]
```

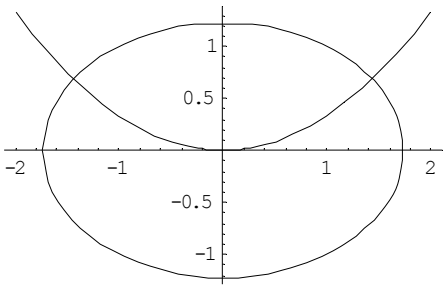



```
Out[2]= - Graphics -
```

```
In[3]:= obr2 = Plot[1 / 3 x^2, {x, -2, 2}]
```




```
Out[3]= - Graphics -
In[4]:= Show[obr1, obr2]
```



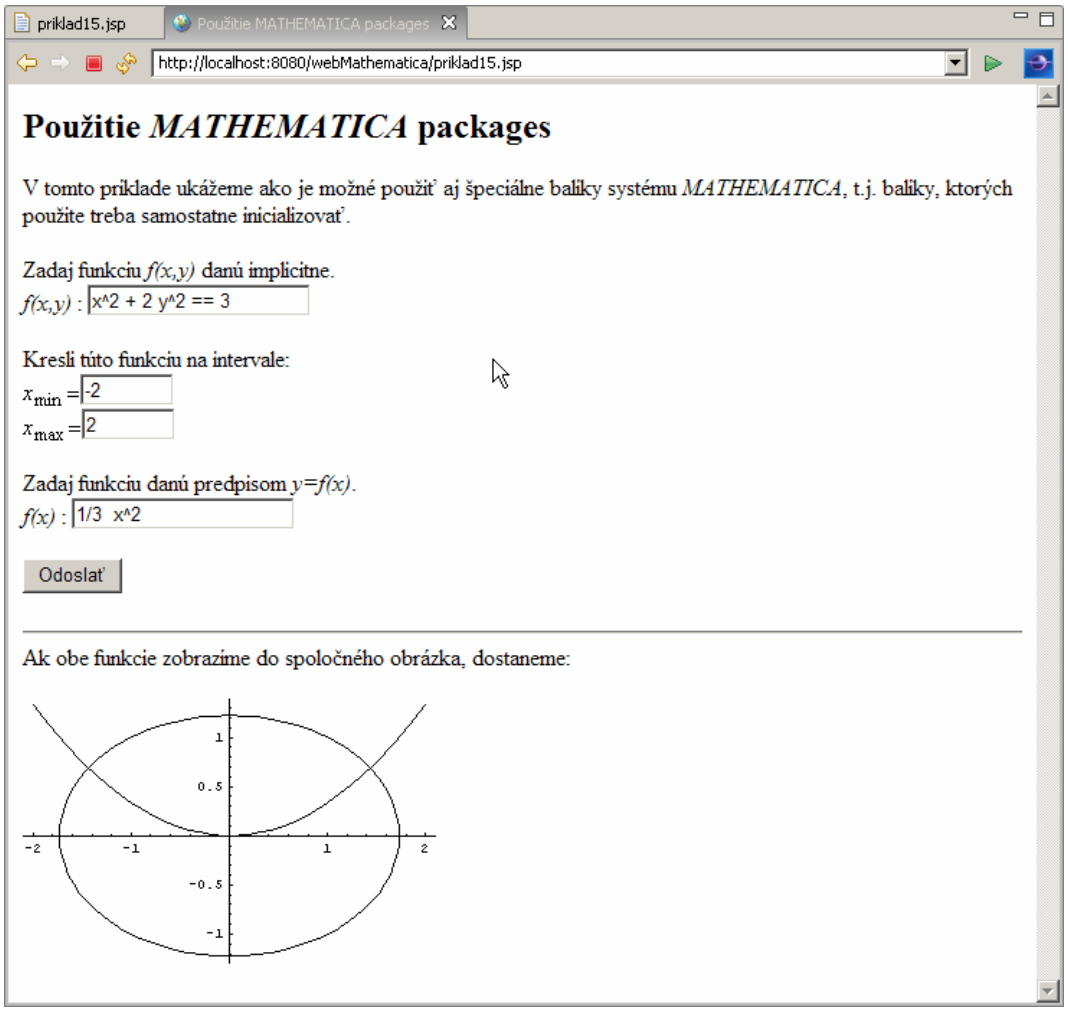

```
Out[4]= - Graphics -
```

Stránka realizujúca túto ukážku bude mať nasledujúci zdrojový kód. Tak ako vo všetkých ostatných príkladoch v tejto kapitole uvádzame len telo jsp stránky.

```
<body bgcolor="#ffffff">
<h2>Použitie <i>MATHEMATICA</i> packages</h2>

<p>V tomto príklade ukážeme ako je možné použiť aj špeciálne balíky systému
<i>MATHEMATICA</i>,
t.j. balíky, ktorých použite treba samostatne inicializovať.</p>
<msp:allocateKernel>
<msp:evaluate>
   MSPPageOptions[ "ContentType" -> "text/xml"];
   Needs["Graphics`ImplicitPlot`"]
</msp:evaluate>

<form action="priklad15.jsp" method="post">

<p>Zadaj funkciu <i>f(x,y) </i> danú implicitne. <br/>
<i>f(x,y) </i>: <input type="text" name="fun1" size="20"
            value="<msp:evaluate> MSPValue[$$fun1,"x^2 + 2 y^2 == 3"]
</msp:evaluate>" />
<msp:evaluate>
If[MSPValueQ[$$fun1],
            fun1=ToExpression[$$fun1],
            fun1 = ToExpression["x^2 + 2 y^2 == 3"] ];
</msp:evaluate>
<br/><br/>

Kresli túto funkciu na intervale:<br/>
<math xmlns='http://www.w3.org/1998/Math/MathML'
    mathematica:form='StandardForm'
    xmlns:mathematica='http://www.wolfram.com/XML/'>
 <mrow>
```

```
  <mtext> </mtext>
  <mrow>
   <msub>
    <mi>x</mi>
    <mi>min</mi>
   </msub>
   <mo>=</mo>
  </mrow>
 </mrow>
</math> <input type="text" name="xmin1" size="5"
      value="<msp:evaluate> MSPValue[$$xmin1,"-2"] </msp:evaluate>" />
<msp:evaluate>
      If[MSPValueQ[$$xmin1], xmin1=ToExpression[$$xmin1], xmin1 = -2 ];
</msp:evaluate> <br/>

<math xmlns='http://www.w3.org/1998/Math/MathML'
    mathematica:form='StandardForm'
    xmlns:mathematica='http://www.wolfram.com/XML/'>
 <mrow>
  <mtext> </mtext>
  <mrow>
   <msub>
    <mi>x</mi>
    <mi>max</mi>
   </msub>
   <mo>=</mo>
  </mrow>
 </mrow>
</math> <input type="text" name="xmax1" size="5"
            value="<msp:evaluate> MSPValue[$$xmax1,"2"] </msp:evaluate>" />
<msp:evaluate>
      If[MSPValueQ[$$xmax1], xmax1=ToExpression[$$xmax1], xmax1 = 2 ];
</msp:evaluate>
</p>

<p>Zadaj funkciu dan[ predpisom <i>y=f(x)</i>. <br/>
<i>f(x) </i>: <input type="text" name="fun2" size="20"
            value="<msp:evaluate> MSPValue[$$fun2,"1/3  x^2"]
</msp:evaluate>" />
<msp:evaluate>
If[MSPValueQ[$$fun2],
            fun2=ToExpression[$$fun2],
            fun2 = ToExpression["1/3  x^2"] ];
</msp:evaluate><br/>
</p>

<input type="submit" name="tlacidlo" value="Odoslať" />
</form>
<hr/>

Ak obe funkcie zobrazíme do spoločného obrázka, dostaneme: <br/><br/>
<msp:evaluate>
      obr1 = ImplicitPlot[fun1, {x, xmin1, xmax1}];
      obr2 = Plot[fun2, {x, xmin1, xmax1}];
      MSPShow[Show[obr1, obr2]]
</msp:evaluate>

</msp:allocateKernel>
</body>
```

Po zobrazení stránky dostaneme výsledok veľmi podobný výsledku výpočtu v systéme *MATHEMATICA*.



Vytvorená stránka je samozrejme plne funkčná a užívateľ môže zadať vlastnú implicitnú funkciu, interval pre kreslenie aj funkciu danú predpisom $y = f(x)$. Tento príklad neobsahuje žiadne ďalšie nové myšlienky okrem ukážky, že je možné spojiť grafické objekty vytvorené dvomi principiálne rôznymi grafickými príkazmi.

Dôležité v prípade takejto kombinácie príkazov je uvedomiť si, že **nie je podstatné akým príkazom sme grafický objekt vytvorili, ale aký typ grafického objektu sme vytvorili**. V našom príklade sme v oboch prípadoch vytvorili grafické objekt typu `-Graphics-`, ktoré je možné ľubovoľne kombinovať pomocou príkazu `Show` a následne zobraziť na stránke pomocou príkazu `MSPShow`.

Rovnako ako v predchádzajúcich príkladoch tejto série pripomíname, že takto vytvorené aplikácia nie je z pedagogického hľadiska správna a kompletná. Pri tvorbe reálnych pedagogických aplikácií je potrebné ošetriť zadávané vstupy na ich syntaktickú správnosť, alebo podrobne vo vstupnom texte vysvetliť, akým spôsobom je potrebné zadať vstupy do vstupného formulára.

5 Programujeme na jsp stránke

V nasledujúcej ukážke uvedieme príklad použitia **web*MATHEMATICE* pri vizualizácii riešenia nerovníc.** *MATHEMATICA* neobsahuje žiadny štandardný príkaz, ktorý by umožňoval vizualizovať riešenie nerovníc (učivo strednej školy, 2 ročník), preto je potrebné jednoduchými príkazmi tento postup simulovať a výsledok výpočtu znázorniť.

Najskôr si vysvetlíme realizáciu takéhoto výpočtu v systéme *MATHEMATICA* a potom uvedieme prepis tohto postupu do jsp stránky.

V *MATHEMATICA* packages (balíku) **`Algebra`** je definovaný aj príkaz na riešenie nerovníc. Tento balík načítame a následne pomocou predefinovaného príkazu **`InequalitySolve`** vyriešime konkrétnu nerovnicu.

```
In[1]:= << Algebra`InequalitySolve`
        nerovnica = InequalitySolve[(x^2 - 5) (x + 7) (x - 3) > 0, x]

Out[2]= x < -7 || -√5 < x < √5 || x > 3
```

Riešením tejto nerovnice je zjednotenie intervalov $(-\infty,-7)\cup(-\sqrt{5},\sqrt{5})\cup(3,\infty)$. Táto interpretácia je výhodná v prípadoch, ak by naším cieľom bolo analyzovať výsledok z hľadiska štruktúry výsledku (pri vytváraní jsp stránok, ktoré budú napríklad step by step riešiť nerovnice), s použitím príkazu **`FullForm`**.

```
In[3]:= FullForm[nerovnica]

Out[3]//FullForm=
Or[Less[x, -7], Inequality[Times[-1, Power[5, Rational[1, 2]]],
  Less, x, Less, Power[5, Rational[1, 2]]], Greater[x, 3]]
```

Ak požadujeme grafický výstup – chceme výsledok riešenia nerovnice znázorniť na číselnej osi, nie je pre nás štruktúra výsledku až tak veľmi podstatná. Všimnite si, že **výsledkom riešenia nerovnice je v skutočnosti logický výraz, u ktorého môžeme vyhodnocovať jeho pravdivosť resp. nepravdivosť pre rôzne hodnoty čísla** x.

```
In[4]:= nerovnica /. x → 2

Out[4]= True

In[5]:= nerovnica /. x → -6

Out[5]= False
```

Ak by sme pre každé $x\in(a,b)$ na ktorom budeme výsledok znázorňovať vyhodnotili, či toto x je riešením nerovnice, môžeme výsledok znázorniť aj bez znalosti štruktúry nerovnice. Nemusíme dokonca vyhodnocovať ani jednotlivé body, ale stačí rozdeliť interval (a,b) na dostatočne malé intervaly a vyhodnocovať platnosť nerovností (logických výrazov) v strede každého takéhoto intervalu. Výsledok znázorňujeme vždy len v istom rozlíšení a veľkosti, a v prípade jsp stránky nemá možnosť užívateľ tieto parametre ovplyvniť. Úplne postačujúce je zvoliť delenie intervalu (a,b) na tak malé intervaly, ktoré budú zodpovedať výslednému použitému rozlíšeniu.

Ak stred tohto malého intervalu spĺňa nerovnosť do výsledného obrázka zaznačíme, že celý interval je riešením nerovnice. Výsledný grafický objekt je teda len zjednotením jednotlivých „čiar“, jednotlivých malých intervalov, v strede ktorých bola nerovnica splnená. Poďme túto

myšlienku naprogramovať. Výsledok samozrejme nie je úplne presný, ale pri vhodnej kombinácii veľkosti výsledného obrázka a počtu deliacich intervalov, nie je táto nepresnosť badateľná.

Najskôr definujeme koncové body intervalu (a,b), na ktorom chceme graficky znázorniť výsledok riešenia tejto nerovnice.

```
In[6]:= min = -10;
        max = 10;
        krok = (max - min) / 40;
```

Vzhľadom na rozlíšenie obrazovky a default nastavenie veľkosti obrázku na 400 px, postačuje tento interval rozdeliť na 40 malých intervalov, na ktorých budeme vyhodnocovať riešenie nerovnice.

Teraz vytvoríme pole, ktoré bude obsahovať výsledok vyhodnotenia logického výrazu – **nerovnica** a interval v ktorého ľavom koncovom bode bola platnosť tejto nerovnice splnená.

```
In[9]:= pom = Table[{nerovnica /. x → i + krok / 2, i, i + krok} // N,
          {i, min, max, krok}]
```

```
Out[9]= {{True, -10., -9.5}, {True, -9.5, -9.}, {True, -9., -8.5},
         {True, -8.5, -8.}, {True, -8., -7.5}, {True, -7.5, -7.},
         {False, -7., -6.5}, {False, -6.5, -6.}, {False, -6., -5.5},
         {False, -5.5, -5.}, {False, -5., -4.5}, {False, -4.5, -4.},
         {False, -4., -3.5}, {False, -3.5, -3.}, {False, -3., -2.5},
         {False, -2.5, -2.}, {True, -2., -1.5}, {True, -1.5, -1.},
         {True, -1., -0.5}, {True, -0.5, 0.}, {True, 0., 0.5},
         {True, 0.5, 1.}, {True, 1., 1.5}, {True, 1.5, 2.}, {False, 2., 2.5},
         {False, 2.5, 3.}, {True, 3., 3.5}, {True, 3.5, 4.}, {True, 4., 4.5},
         {True, 4.5, 5.}, {True, 5., 5.5}, {True, 5.5, 6.}, {True, 6., 6.5},
         {True, 6.5, 7.}, {True, 7., 7.5}, {True, 7.5, 8.}, {True, 8., 8.5},
         {True, 8.5, 9.}, {True, 9., 9.5}, {True, 9.5, 10.}, {True, 10., 10.5}}
```

Zo získaného výsledku môžeme prečítať, že v strede intervalu $[-10,-9.5]$ nerovnica nebola splnená, ale v strede intervalu $[-1,-0.5]$ bola splnená. Pre každý takto vytvorený prvok poľa s hodnotou **True**, potrebujeme teraz vytvoriť grafický objekt (čiaru), ktorú vo výsledku znázorníme. Musíme si definovať vlastnú funkciu, ktorá vytvorenie takéhoto grafického objektu zabezpečí.

```
In[10]:= Clear[test]
         test[{n_, xmin_, xmax_}] :=
          If[n, Graphics[{Blue, Thickness[0.015],
             Line[{{xmin, 0}, {xmax, 0}}]}]]
```

Takto definovanú funkciu aplikujeme na vytvorené pole.

```
In[12]:= pom2 = Map[test, pom]
```

```
Out[12]= {- Graphics -, - Graphics -, - Graphics -, - Graphics -, - Graphics -, - Graphics -,
          Null, Null, Null, Null, Null, Null, Null, Null, Null, Null, - Graphics -,
          - Graphics -, - Graphics -, - Graphics -, - Graphics -, - Graphics -, - Graphics -,
          - Graphics -, Null, Null, - Graphics -, - Graphics -, - Graphics -, - Graphics -,
          - Graphics -, - Graphics -, - Graphics -, - Graphics -, - Graphics -, - Graphics -,
          - Graphics -, - Graphics -, - Graphics -, - Graphics -, - Graphics -}
```

Na každom intervale, kde bol pôvodný logický výraz vyhodnotený ako pravdivý je teraz vytvorený grafický objekt. Do výsledného objektu, ktorý bude reprezentovať riešenie nerovnice vyberieme len vytvorené grafické objekty – objekty typu **-Graphics-**. Pretože sme objekty definovali ako bezrozmerné objekty typu čiara, potrebujeme ešte nakresliť označenú a popísanú číselnú os a oba obrázky spojiť.

```
In[13]:= obr1 = ParametricPlot[{x, 0}, {x, -10, 10},
            PlotStyle → {Black, Thickness[0.007]},
            AspectRatio → 0.2, Axes → {Automatic, None}];
          obr2 = Cases[pom2, _Graphics];
          Show[obr1, obr2]
```


```
Out[15]= - Graphics -
```

Vidíme, že výsledný obrázok skutočne reprezentuje grafické znázornenie riešenia nerovnice na intervale $(-10,10)$. Teraz poďme vytvoriť jsp stránku, ktorá bude realizovať rovnaký výpočet. Užívateľovi stránky musíme umožniť zadať nerovnicu, ktorú chce riešiť a interval, na ktorom budeme znázorňovať výsledok riešenia nerovnice.

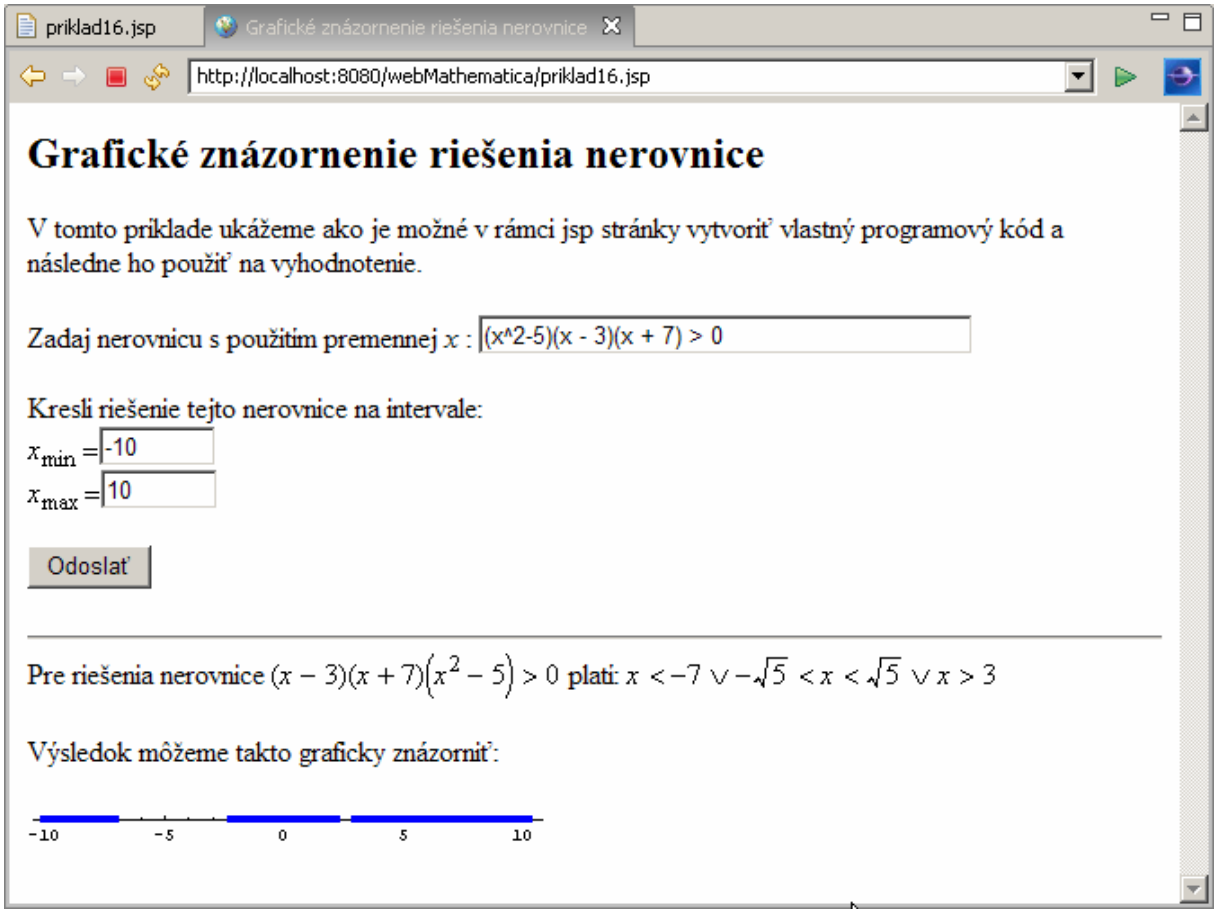
Zdrojový kód tejto stránky má nasledujúci tvar. . Tak ako vo všetkých ostatných príkladoch v tejto kapitole uvádzame len telo jsp stránky.

```
<body bgcolor="#ffffff">
<h2>Grafické znázornenie riešenia nerovnice</h2>

<p>V tomto príklade ukážeme ako je možné v rámci jsp stránky
vytvoriť vlastný programový kód a následne ho použiť na vyhodnotenie.</p>

<msp:allocateKernel>
<msp:evaluate>
   MSPPageOptions[ "ContentType" -> "text/xml"];
   Needs["Algebra`InequalitySolve`"]
</msp:evaluate>


<form action="priklad16.jsp" method="post">

<p>Zadaj nerovnicu s použitím premennej <i> x </i>:

<input type="text" name="nerovnica" size="40"
            value="<msp:evaluate>
                       MSPValue[$$nerovnica,"(x^2-5)(x - 3)(x + 7) > 0"]
                   </msp:evaluate>" />
<msp:evaluate>
If[MSPValueQ[$$nerovnica],
            nerovnica=ToExpression[$$nerovnica],
            nerovnica = ToExpression["(x^2-5)(x - 3)(x + 7) > 0"] ];
</msp:evaluate>
</p>

<p>
Kresli riešenie tejto nerovnice na intervale:<br/>
```

```
<math xmlns='http://www.w3.org/1998/Math/MathML'
    mathematica:form='StandardForm'
    xmlns:mathematica='http://www.wolfram.com/XML/'>
 <mrow>
  <mtext> </mtext>
  <mrow>
   <msub>
    <mi>x</mi>
    <mi>min</mi>
   </msub>
   <mo>=</mo>
  </mrow>
 </mrow>
</math> <input type="text" name="min" size="5"
            value="<msp:evaluate> MSPValue[$$min,"-10"] </msp:evaluate>" />
<msp:evaluate>
      If[MSPValueQ[$$min], min=ToExpression[$$min], min = -10 ];
</msp:evaluate> <br/>

<math xmlns='http://www.w3.org/1998/Math/MathML'
    mathematica:form='StandardForm'
    xmlns:mathematica='http://www.wolfram.com/XML/'>
 <mrow>
  <mtext> </mtext>
  <mrow>
   <msub>
    <mi>x</mi>
    <mi>max</mi>
   </msub>
   <mo>=</mo>
  </mrow>
 </mrow>
</math> <input type="text" name="max" size="5"
            value="<msp:evaluate> MSPValue[$$max,"10"] </msp:evaluate>" />
<msp:evaluate>
      If[MSPValueQ[$$max], max=ToExpression[$$max], max = 10 ];
</msp:evaluate>
</p>

<input type="submit" name="tlacidlo" value="Odoslať" />
</form>
<hr/>

Pre riešenia nerovnice
<msp:evaluate>
      MSPFormat[nerovnica, MathMLForm]
</msp:evaluate>
platí:
<msp:evaluate>
      riesenie = InequalitySolve[nerovnica//N, x];
      MSPFormat[riesenie, MathMLForm]
</msp:evaluate>

<br/>
<br/>
Výsledok môžeme takto graficky znázorniť:<br/>
<msp:evaluate>
      krok = (max - min)/80;
      pom = Table[{riesenie/. x -> i + krok/2, i, i +
          krok} // N, {i, min, max, krok}];
```

```
    Clear[test];
      test[{n_, xmin_, xmax_}] := If[n, Graphics[{Blue, Thickness[0.015],
                      Line[{{xmin, 0}, {xmax, 0}}]}]];
      pom2 = Map[test, pom];

      obr1 = ParametricPlot[{x, 0}, {x, min, max},
                  PlotStyle -> {Black, Thickness[0.007]}, AspectRatio ->
0.2,
                  Axes -> {Automatic, None}];
      obr2 = Cases[pom2, _Graphics];
      MSPShow[Show[obr1, obr2]]
</msp:evaluate>

</msp:allocateKernel>

</body>
```

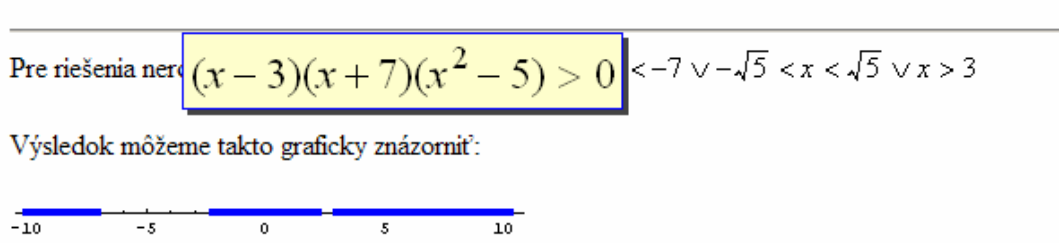
Po zobrazení stránky dostaneme výsledok veľmi podobný výsledku výpočtu v systéme *MATHEMATICA*.



Všimnite si, že vytvorený zdrojový kód jsp stránky sa od pôvodnej ukážky navrhnutej pre systém *MATHEMATICA* odlišuje len v detailoch: zvolili sme väčší počet deliacich intervalov. Výsledok bude pri tejto voľbe presnejší. Všimnite si zobrazenie nerovnice a riešenia nerovnica na stránke – čitateľné a prehľadné zobrazenie nám zabezpečuje použitie MATHML kódu.

Poďme teraz analyzovať zdrojový kód tejto stránky. Najskôr načítame všetky balíky (packages) potrebné pre realizáciu výpočtu. V prvej časti zdrojového kódu – vo formulári používame len štandardný spôsob definície vstupných premenných. Umožníme užívateľovi stránky zadať nerovnicu, ktorú chce riešiť a interval, na ktorom chce graficky znázorniť výsledok.

Pod čiarou, ktorá na výslednej stránke oddeľuje formulár a časť výsledkov zobrazíme nerovnicu, ktorú chceme riešiť a výsledok jej riešenia tak, ako nám ho poskytol príkaz na riešenie nerovníc **`InequalitySolve`**. Všimnite si čitateľnosť zobrazených výsledkov. Je dôsledkom použitia MathML kódu. Matematický text kódovaný touto formou je živým textom, napríklad po kliknutím na takýto objekt sa nám matematický vzťah zväčší – čo automaticky zlepší jeho čitateľnosť.



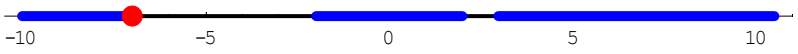
V závere stránky sme do jedného výpočtového prostredia umiestnili celý zdrojový kód, ktorý zrealizuje náš výpočet a výsledkom ktorého je grafické znázornenie riešenia nerovnice. Jednotlivé príkazy použité v tejto časti sa týkajú programovania v systéme *MATHEMATICA* a boli podrobne vysvetlené v predchádzajúcej časti.

Na záver pripomeňme, že pre reálne použitie je potrebné túto stránku ešte vylepšiť napríklad pridaním vykreslenia krajných bodov intervalu a zdôraznením, či krajné body patria alebo nepatria do množiny riešení tejto nerovnice.

Tento problém je z programátorského hľadiska zložitejší, pretože vyžaduje analýzu štruktúry nerovnice tak, ako sme o tom hovorili v predchádzajúcej časti. Obsahuje však niekoľko zaujímavých myšlienok, ktoré je možné využiť aj pri iných aplikáciách a preto si ho aspoň v hrubých rysoch naznačíme.

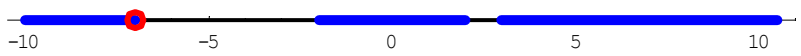
Najskôr musíme vytvoriť a definovať grafické objekty, ktoré budeme používať pri zobrazovaní krajných bodov intervalov. Označíme si ich **`obr3`** a **`obr4`**, pretože **`obr1`** a **`obr2`** sme už použili v predchádzajúcom zdrojovom kóde. Grafický objekt **`obr3`** bude znázorňovať krajný bod intervalu, ktorý do intervalu patrí, grafický objekt **`obr4`** bude znázorňovať krajný bod intervalu, ktorý do intervalu nepatrí.

```
In[16]:= obr3 = Graphics[{Red, PointSize[0.03], Point[{-7, 0}]}]

Out[16]= - Graphics -

In[17]:= obr4 = Graphics[{Red, Thickness[0.01], Circle[{-7, 0}, Offset[{4, 4}]]}]

Out[17]= - Graphics -

In[18]:= Show[obr1, obr2, obr3, ImageSize → {400, Automatic}]
```




```
Out[18]= - Graphics -
```

In[19]:= **Show[obr1, obr2, obr4, ImageSize → {400, Automatic}]**



Out[19]= - Graphics -

Teraz poďme pristúpiť ku analýze štruktúry výsledku riešenia nerovnice, ktorý poskytne príkaz **InequalitySolve.** Použijeme nerovnicu z predchádzajúceho príkladu.

In[20]:= **nerovnica**

Out[20]= $x < -7 \;||\; -\sqrt{5} < x < \sqrt{5} \;||\; x > 3$

Vidíme, že vo výsledku sa môžu vyskytnúť typovo tri rôzne druhy jednoduchých nerovností, a to $x < -7$ $x < číslo$, $x > číslo$, $číslo1 < x < číslo2$. Znamienka v nerovnostiach môžu byť ostré alebo neostré. Aj s nerovnicou môžeme manipulovať rovnako ako s akýmkoľvek výrazom v *MATHEMATICE*. Čitateľovi odporúčame preštudovať si kapitolu o programovaní, alebo publikácie venované programovaniu v systéme *MATHEMATICA*. Jednotlivé základné rovnosti môžeme vybrať pomocou manipulácie s časťami celej nerovnice, napríklad:

In[21]:= **nerovnica[[1]]**

Out[21]= $x < -7$

In[22]:= **Table[nerovnica[[i]], {i, 1, Length[nerovnica]}]**

Out[22]= $\{x < -7, -\sqrt{5} < x < \sqrt{5}, x > 3\}$

Vieme samozrejme zistiť, aj koľko základných nerovností tvorí celkové riešenie:

In[23]:= **Length[nerovnica]**

Out[23]= 3

Poďme sa teraz pozrieť na prvú časť riešenie – nerovnosť $x < -7$. Je tvorená nasledujúcimi časťami:

In[24]:= **cast = nerovnica[[1]]**

Out[24]= $x < -7$

In[25]:= **cast[[1]]**

Out[25]= x

In[26]:= **cast[[2]]**

Out[26]= -7

In[27]:= **Head[cast]**

Out[27]= Less

Vo všeobecnosti **Head [ ]** môže nadobúdať nasledujúce hodnoty

In[28]:= **Head[x < 7]**

Out[28]= Less

In[29]:= **Head[x <= 7]**

Out[29]= LessEqual

In[30]:= **Head[x >= 7]**

Out[30]= GreaterEqual

In[31]:= **Head[x > 7]**

Out[31]= Greater

Musíme si ešte uvedomiť, že nerovnica základného typu môže byť zapísaná buď v tvare $x < -7$, alebo $-7 < x$. Preto je potrebné ošetriť aj otázku správneho výberu číselnej časti tejto nerovnice. Napríklad definíciou takejto funkcie:

```
In[32]:= If[NumberQ[cast[[2]]], bod = cast[[2]], bod = cast[[1]]]

Out[32]= -7
```

Potom už môžeme definovať grafický objekt požadovaných vlastností a znázorniť ho spolu s predtým definovanými obrázkami (číselnou osou a intervalmi).

```
In[33]:= obr = Which[
            Head[cast] === Less,
          Graphics[{Red, Thickness[0.01], Circle[{bod, 0}, Offset[{4, 4}]]}],
            Head[cast] === LessEqual,
          Graphics[{Red, PointSize[0.03], Point[{bod, 0}]}],
            Head[cast] === Greater,
          Graphics[{Red, Thickness[0.01], Circle[{bod, 0}, Offset[{4, 4}]]}],
            Head[cast] === GreaterEqual,
          Graphics[{Red, PointSize[0.03], Point[{bod, 0}]}]
           ]

In[34]:= Show[obr1, obr2, obr]
```


```
Out[34]= - Graphics -
```

Rovnaký postup uplatníme, ak budeme analyzovať poslednú časť riešenia. Nebudeme ju tu uvádzať, čitateľ ju nájde v súboroch, ktoré sú prílohou tejto knihy.

Stredná časť je typovo odlišná, predstavuje nerovnosti v tvare $číslo1 < x < číslo2$.

```
In[42]:= cast = nerovnica[[2]]

Out[42]= -√5 < x < √5
```

Štruktúra tohto typu nerovnice je nasledujúca:

```
In[43]:= Length[cast]

Out[43]= 5

In[44]:= Table[cast[[i]], {i, 1, Length[cast]}]

Out[44]= {-√5, Less, x, Less, √5}
```

Číselné hodnoty, ktoré potrebujeme znázorniť musíme vybrať ako prvú a poslednú časť nerovnosti a pre každý z týchto bodov musíme definovať príslušný grafický objekt.

```
In[45]:= If[Length[cast] > 2, bod1 = First[cast]; bod2 = Last[cast];]
```

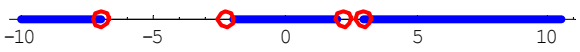
Z programátorského hľadiska sa ukazuje výhodné definovať funkciu **koliesko**, ktorá bude pre zadané parametre vykresľovať príslušný grafický objekt. Túto funkciu budeme používať aj pri zobrazovaní riešenia jednoduchý nerovností, aj pri zobrazovaní riešenia úplných nerovností.

Jedna z možností, ako takúto funkciu definovať nasleduje:

```
In[46]:= koliesko[bod_, typ_] := Which[
              typ === Less, Graphics[{Red, Thickness[0.01], Circle[{bod, 0}, Offset[{4, 4}]]}],
              typ === LessEqual, Graphics[{Red, PointSize[0.03], Point[{bod, 0}]}],
              typ === Greater, Graphics[{Red, Thickness[0.01], Circle[{bod, 0}, Offset[{4, 4}]]}],
              typ === GreaterEqual, Graphics[{Red, PointSize[0.03], Point[{bod, 0}]}]
                 ]
```

Takto definovanú funkciu využijeme pri zostavený globálnej postupnosti príkazov pre vyriešenie nášho problému.

```
In[47]:= obr = {};
      For[i = 1, i ≤ Length[nerovnica], i++,
       cast = nerovnica[[i]];
       If[Length[cast] == 2,
            bod = cast[[2]];
            typ = Head[cast];
            obr = Append[obr, koliesko[bod, typ]]
        ]
        If[Length[cast] == 5,
            bod1 = cast[[1]]; typ1 = cast[[2]];
            bod2 = Last[cast]; typ2 = cast[[-2]];
         obr = Append[obr, koliesko[bod1, typ1]];
         obr = Append[obr, koliesko[bod2, typ2]];
        ]
      ]

In[49]:= obr

Out[49]= {- Graphics -, - Graphics -, - Graphics -, - Graphics -}

In[50]:= Show[obr1, obr2, obr]
```



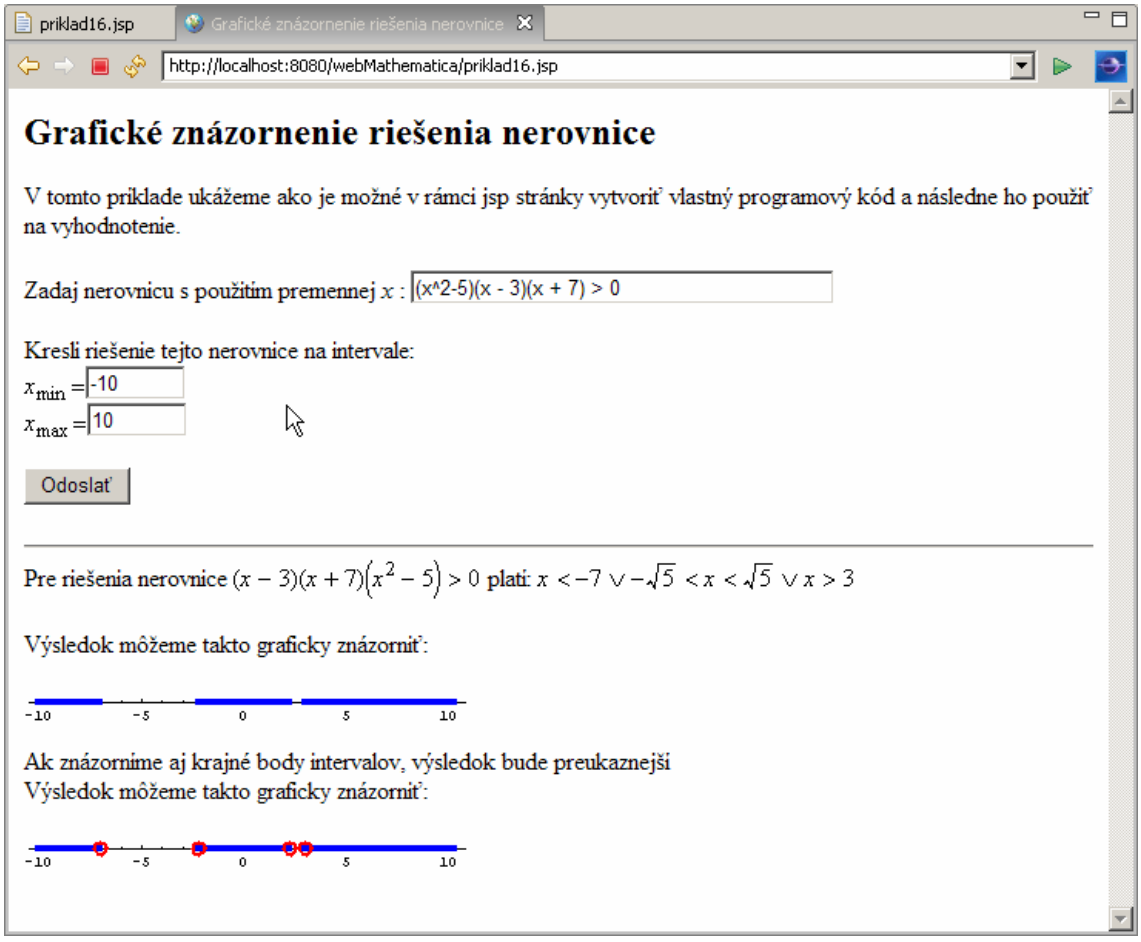

```
Out[50]= - Graphics -
```

Postupne prehľadáme celú štruktúru nerovnice a vytvoríme postupnosť príslušných grafických objektov, ktoré následne zobrazíme do výsledného grafu.

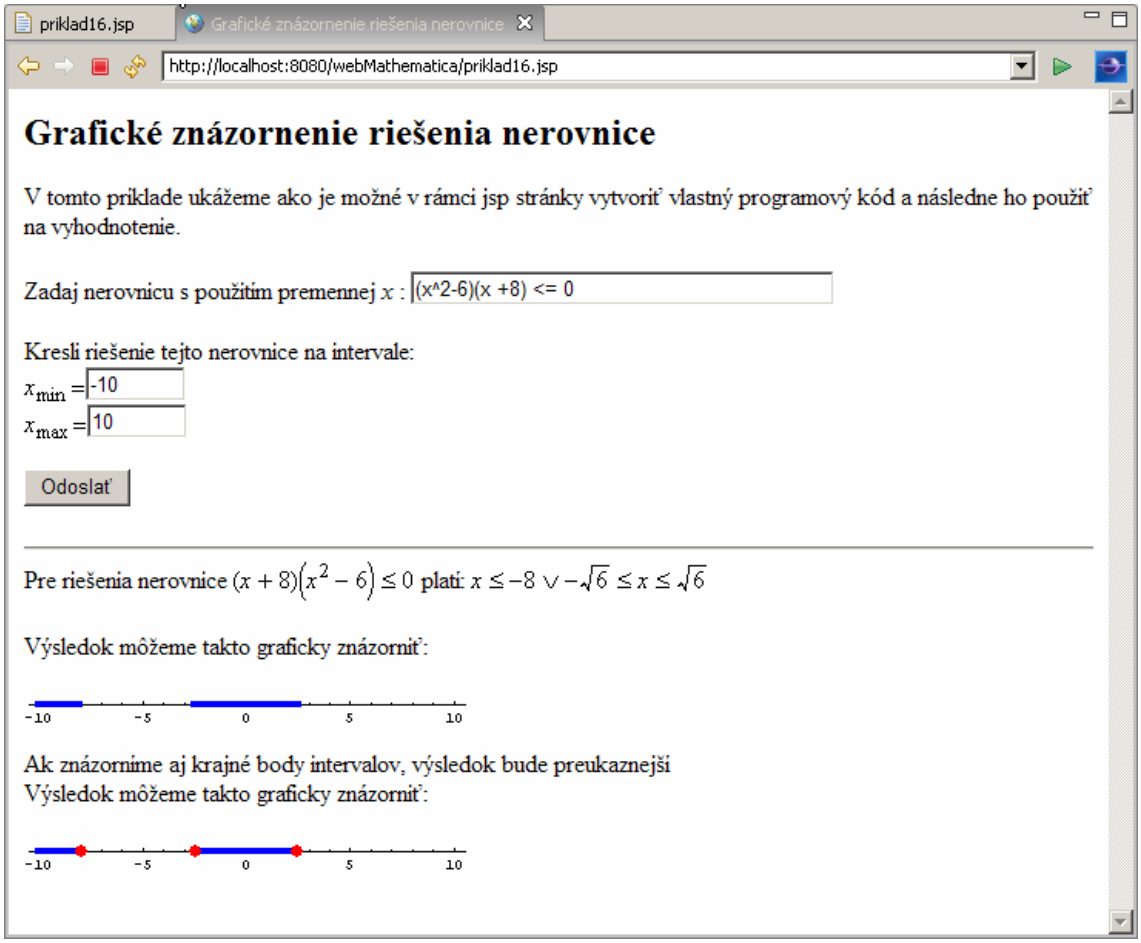
Takto pripravený programový kód môžeme umiestniť do pôvodnej jsp stránky a otestovať jeho funkčnosť. Do pôvodného výpočtového kódu na koniec doplníme:

```
.....
      obr2 = Cases[pom2, _Graphics];
      MSPShow[Show[obr1, obr2]]
</msp:evaluate>

<br/>
Ak znázorníme aj krajné body intervalov, výsledok bude preukaznejší<br/>
Výsledok môžeme takto graficky znázorniť:<br/>
<msp:evaluate>
koliesko[bod_, typ_] :=
Which[
```

```
      typ === Less,
      Graphics[{Red, Thickness[0.01], Circle[{bod, 0}, Offset[{4, 4}]]}],
      typ === LessEqual, Graphics[{Red, PointSize[0.03], Point[{bod, 0}]}],
      typ === Greater,
        Graphics[{Red, Thickness[0.01], Circle[{bod, 0}, Offset[{4, 4}]]}],
      typ === GreaterEqual,
        Graphics[{Red, PointSize[0.03], Point[{bod, 0}]}]
      ];

obr = {};

For[i = 1, i ≤ Length[riesenie], i++,
  cast = riesenie[[i]];
  If[Length[cast] == 2,
            bod = cast[[2]];
            typ = Head[cast];
            obr = Append[obr, koliesko[bod, typ]]
      ];
    If[Length[cast] == 5,
            bod1 = cast[[1]]; typ1 = cast[[2]];
            bod2 = Last[cast]; typ2 = cast[[-2]];
      obr = Append[obr, koliesko[bod1, typ1]];
      obr = Append[obr, koliesko[bod2, typ2]];
      ];
      ];
  MSPShow[Show[obr1, obr2, obr]]
</msp:evaluate>
```



Funkčnosť vytvorenej jsp stránky môžeme otestovať aj na inom type nerovnice:



Samozrejme, že uvedený programový kód nie je kompletný, jeho funkčnosť závisí od schopností príkazu na riešenie nerovníc **`InequalitySolve`**. Z pedagogického hľadiska by bolo možné doplniť do jsp stránky aj ukážku postupného riešenia nerovnice a znázorňovania výsledku do grafu, alebo podrobnejšie vysvetlenie. Cieľom tohto príkladu nebolo úplne spracovať daný problém, ale poskytnúť čitateľovi ukážku, ako je možné integrovať skúsenosti z programovania v *MATHEMATICE* do vytváraných jsp stránok a zároveň ukážku, že v mnohých prípadoch web*MATHEMATICA* je skutočne len komunikačným nástrojom medzi užívateľom stránky a výpočtovým jadrom *MATHEMATICE*.