

DOKUMENTY S GRAFICKÝM OBSAHEM

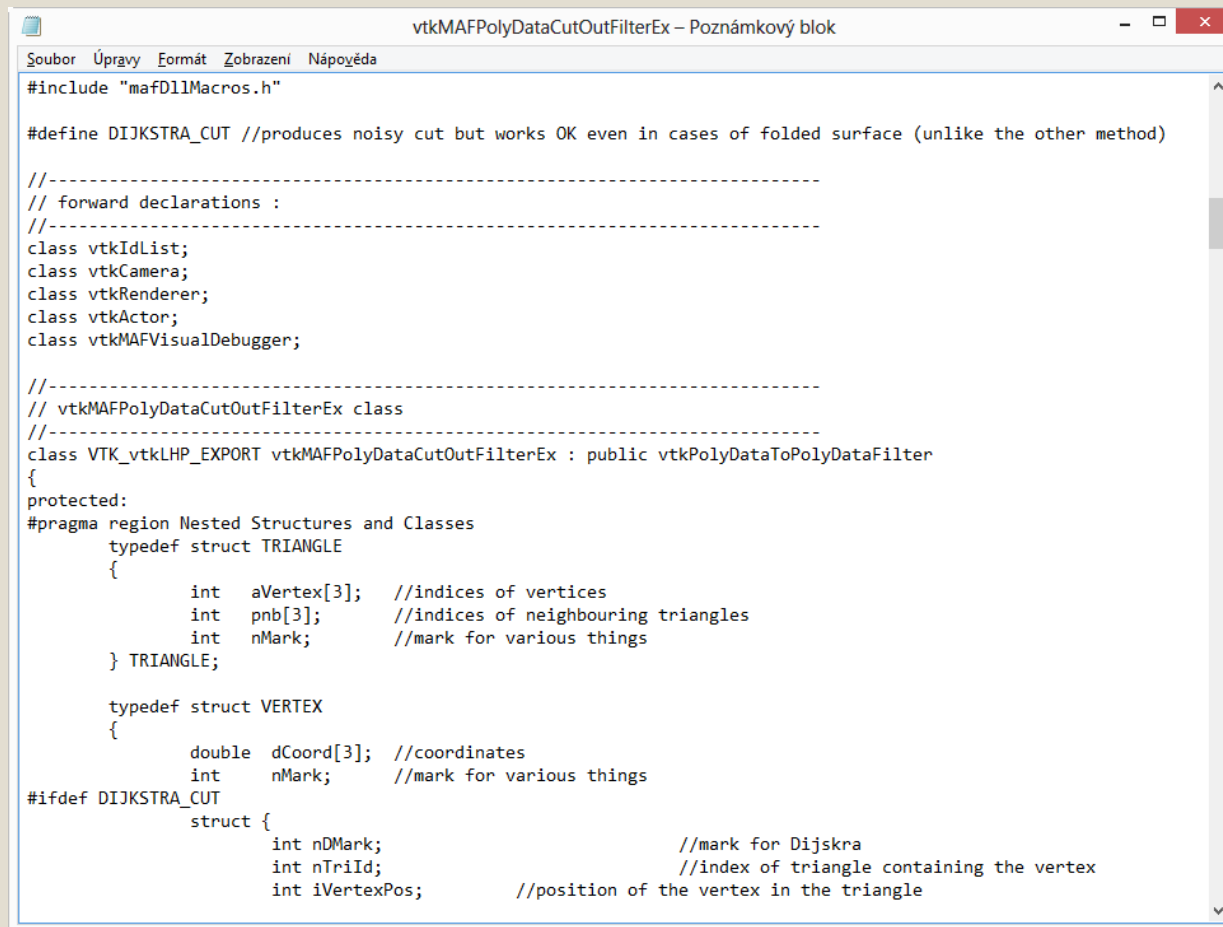
Vyznačovací
jazyky

Vzhled a
rozložení
stránky

TEXTOVÝ DOKUMENT

- Textový soubor (řetězec)
 - Např. zdrojové kódy, soubory .txt, ...
- Obsahuje
 - Odstavce oddělené „koncem“ odstavce
 - Např. znak pro <ENTER>
 - Věty
 - Slova
- Vizualizace obsahu závislá na aplikaci, ve které dokument se otevírá

TEXTOVÝ DOKUMENT



```
Soubor  Úpravy  Formát  Zobrazení  Nápověda

#include "maFDllMacros.h"

#define DIJKSTRA_CUT //produces noisy cut but works OK even in cases of folded surface (unlike the other method)

//-----
// forward declarations :
//-----
class vtkIdList;
class vtkCamera;
class vtkRenderer;
class vtkActor;
class vtkMAFVisualDebugger;

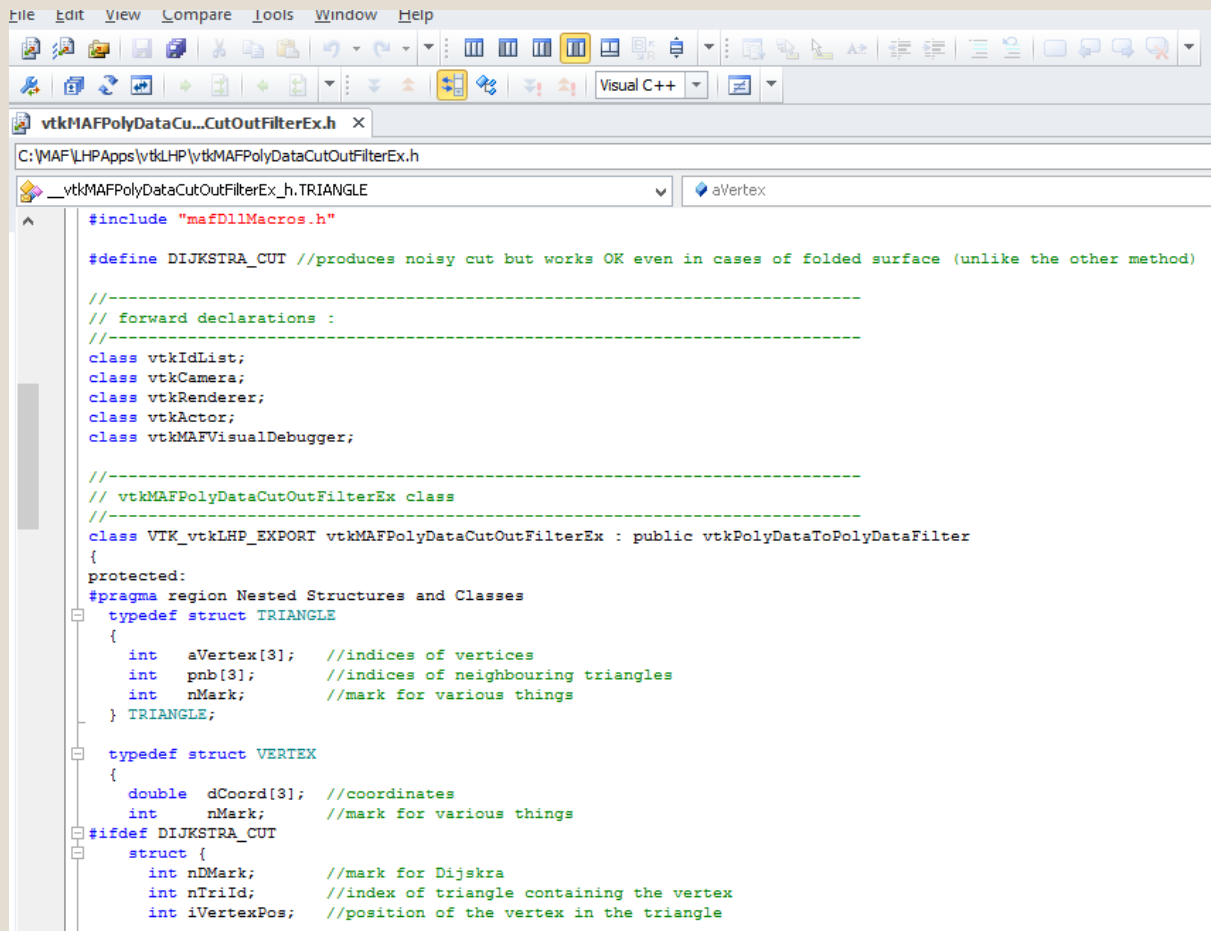
//-----
// vtkMAFPolyDataCutOutFilterEx class
//-----
class VTK_vtkLHP_EXPORT vtkMAFPolyDataCutOutFilterEx : public vtkPolyDataToPolyDataFilter
{
protected:
#pragma region Nested Structures and Classes
    typedef struct TRIANGLE
    {
        int    aVertex[3];    //indices of vertices
        int    pnb[3];        //indices of neighbouring triangles
        int    nMark;         //mark for various things
    } TRIANGLE;

    typedef struct VERTEX
    {
        double dCoord[3];    //coordinates
        int    nMark;        //mark for various things
    } VERTEX;

#pragma endregion

#ifdef DIJKSTRA_CUT
    struct {
        int nDMark;           //mark for Dijskra
        int nTriId;           //index of triangle containing the vertex
        int iVertexPos;       //position of the vertex in the triangle
    } DijkstraData;
#endif
};
```

TEXTOVÝ DOKUMENT



```
file edit view compare tools window help
[Icons]
Visual C++
vtkMAFPolyDataCu...CutOutFilterEx.h x
C:\MAF\LHPApps\vtkLHP\vtkMAFPolyDataCutOutFilterEx.h
__vtkMAFPolyDataCutOutFilterEx_h.TRIANGLE aVertex

#include "mafDllMacros.h"

#define DIJKSTRA_CUT //produces noisy cut but works OK even in cases of folded surface (unlike the other method)

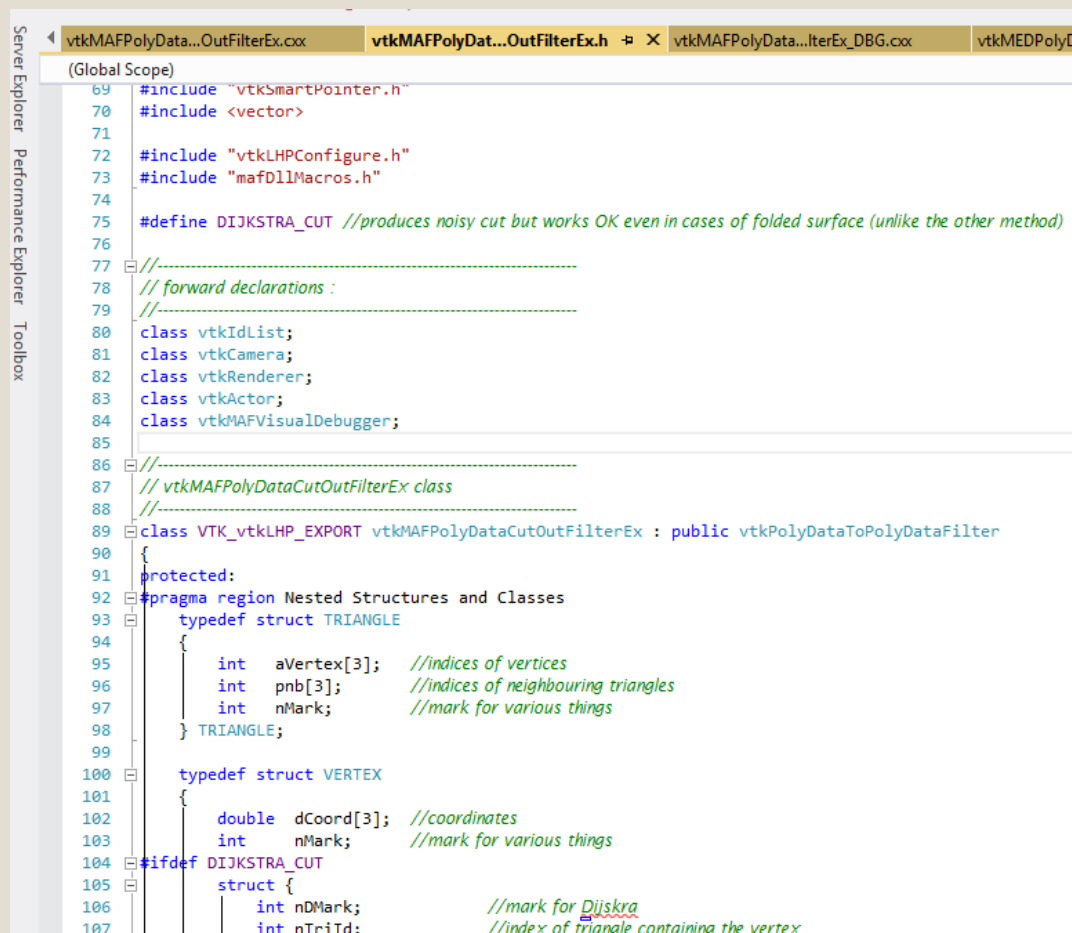
//-----
// forward declarations :
//-----
class vtkIdList;
class vtkCamera;
class vtkRenderer;
class vtkActor;
class vtkMAFVisualDebugger;

//-----
// vtkMAFPolyDataCutOutFilterEx class
//-----
class VTK_vtkLHP_EXPORT vtkMAFPolyDataCutOutFilterEx : public vtkPolyDataToPolyDataFilter
{
protected:
#pragma region Nested Structures and Classes
    typedef struct TRIANGLE
    {
        int    aVertex[3];    //indices of vertices
        int    pnb[3];        //indices of neighbouring triangles
        int    nMark;         //mark for various things
    } TRIANGLE;

    typedef struct VERTEX
    {
        double dCoord[3];    //coordinates
        int    nMark;        //mark for various things
    } VERTEX;

#ifdef DIJKSTRA_CUT
    struct {
        int nDMark;          //mark for Dijskra
        int nTriId;          //index of triangle containing the vertex
        int iVertexPos;      //position of the vertex in the triangle
    }
#endif
};
```

TEXTOVÝ DOKUMENT

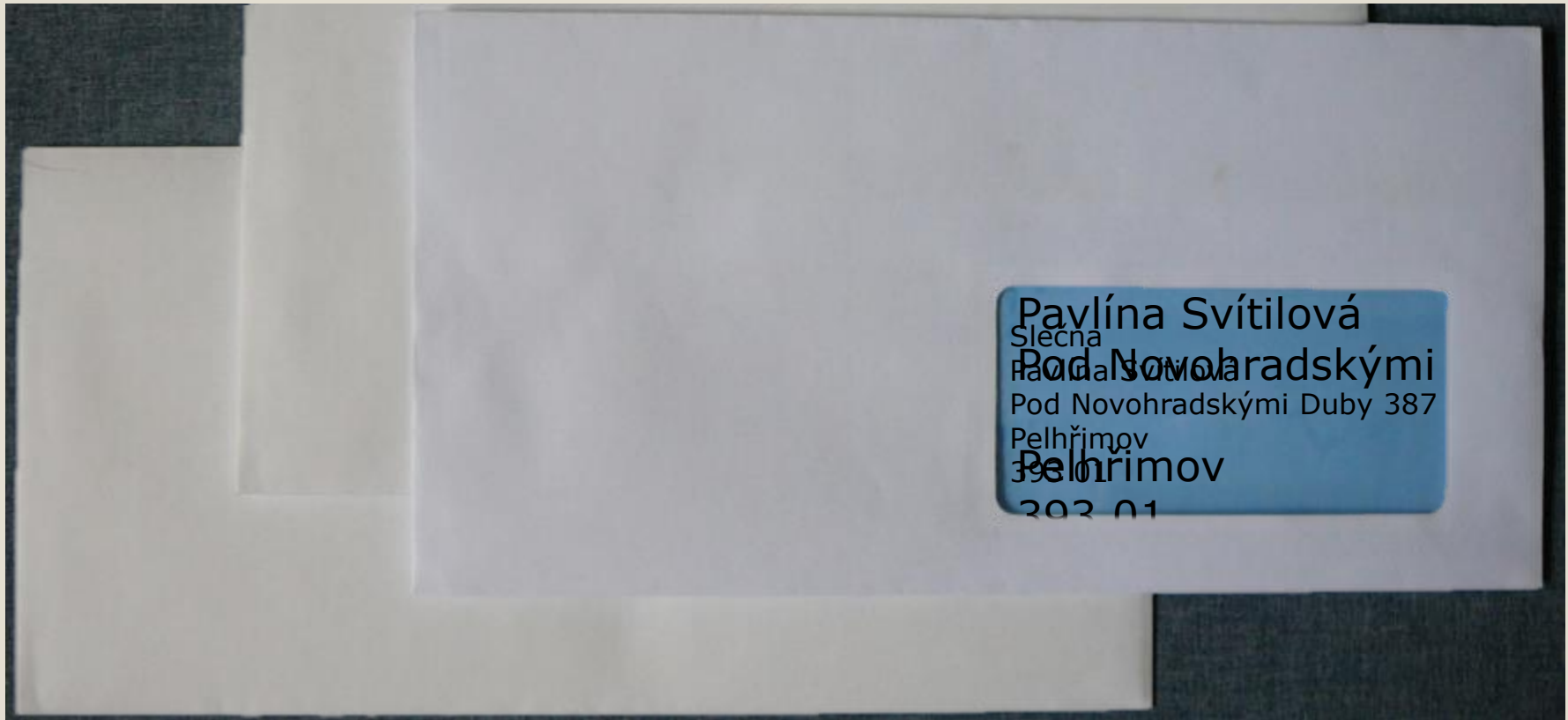


The image shows a screenshot of a C++ source code file, `vtkMAFPolyData...OutFilterEx.h`, within a development environment. The interface includes a 'Server Explorer' on the left, a 'Performance Explorer' in the middle, and a 'Toolbox' on the right. The code is displayed in a central window with a yellow background. It features various preprocessor directives, class declarations, and struct definitions. Comments are present throughout the code, providing context for the implementation. The code is organized into sections separated by dashed lines. The left sidebar shows a tree view with 'Server Explorer', 'Performance Explorer', and 'Toolbox'.

```
(Global Scope)
69 #include "vtkSmartPointer.h"
70 #include <vector>
71
72 #include "vtkLHPConfigure.h"
73 #include "mafDllMacros.h"
74
75 #define DIJKSTRA_CUT //produces noisy cut but works OK even in cases of folded surface (unlike the other method)
76
77 //-----
78 // forward declarations :
79 //-----
80 class vtkIdList;
81 class vtkCamera;
82 class vtkRenderer;
83 class vtkActor;
84 class vtkMAFVisualDebugger;
85
86 //-----
87 // vtkMAFPolyDataCutOutFilterEx class
88 //-----
89 class VTK_vtkLHP_EXPORT vtkMAFPolyDataCutOutFilterEx : public vtkPolyDataToPolyDataFilter
90 {
91 protected:
92 #pragma region Nested Structures and Classes
93     typedef struct TRIANGLE
94     {
95         int    aVertex[3]; //indices of vertices
96         int    pnb[3];    //indices of neighbouring triangles
97         int    nMark;     //mark for various things
98     } TRIANGLE;
99
100     typedef struct VERTEX
101     {
102         double dCoord[3]; //coordinates
103         int    nMark;     //mark for various things
104     } VERTEX;
105 #ifndef DIJKSTRA_CUT
106     struct {
107         int nDMark; //mark for Dijkstra
108         int nTriId; //index of triangle containing the vertex
```

TEXTOVÝ DOKUMENT

- Proč to může vadit?



DOKUMENT S GRAFICKÝM OBSAHEM

- Obsah dokumentu vypadá všude „stejně“
 - Tak, jak bylo zamýšleno tvůrcem obsahu
- Textový obsah doplněn o informaci, jak se má tento obsah zobrazit
 - Např. jakým písmem, kolik znaků na řádek, vizuální oddělení odstavců, stránkování pro tisk, ...
- Textový obsah může být také doplněn o:
 - Vizuální prvky netextového charakteru
 - 2D vektorová nebo rastrová grafika, multimédia
 - Prvky pro interaktivní manipulaci s obsahem

DOKUMENT S GRAFICKÝM OBSAHEM

- Dva základní přístupy:
 - Přiřazení atributů jednotlivým znakům
 - Přiřazení atributů logickým blokům
 - Vyžaduje nadefinování logické struktury dokumentu
 - Např. odstavcům, nadpisům, ...
- Přístupy většinou kombinovány
 - Popsat logickou strukturou nelze vždy 100%
 - Např. v odstavci použito řecké písmeno ε nebo symbol měny €, přičemž tyto glyfy nejsou v požadovaném písmu

PŘIŘAZENÍ ATRIBUTŮ JEDNOTLIVÝM ZNAKŮM

- Jedná se o tzv. atributovaný text
 - Označován též jako „rich text“ nebo „tagovaný“ text
- Vhodné pro krátké jednoúčelové texty
- Atributem barva, podtržení, ...
- Atribut může být:
 - Uložen v pomocné datové struktuře
 - Součástí textového řetězce
- Často vzájemný převod mezi oběma způsoby
 - Např. pro perzistenci obsahu

ATRIBUT V POMOCNÉ STRUKTUŘE

- Atribut uložen obvykle:
 - V seznamu atributů
 - Na zásobníku atributů
- Seznam atributů => atributovaný text reprezentován dvojicí:
 - Neformátovaný (tzv. plain) text
 - Seznam trojic (A, index, počet)
 - A = atribut
 - Index = index prvního znaku, odkud atribut uplatněn
 - Počet = počet znaků, na které uplatnit

ATRIBUT V POMOCNÉ STRUKTUŘE

- Důsledek: lze poskytnout rychle přístup k celému neformátovanému textu
- Při zobrazení řetězce je nezbytné souběžně s vykreslováním znaků procházet seznam
 - Co když nezobrazuji řetězec od jeho začátku?
 - Na problém není univerzální řešení
 - Pro velmi krátké řetězce se většinou neřeší
 - Delší řetězec může být rozdělen na bloky
 - Blokem např. odstavec
 - Každý blok má samostatný seznam
 - Důsledek: obrovský „balast“ v dokumentech
 - Patrné zejména v dokumentech MS aplikací

ATRIBUT V POMOCNÉ STRUKTUŘE

■ Např. Java – třída `AttributedString`

- `AttributedString text = new AttributedString("Pokus");`

```
text.addAttribute(  
    TextAttribute.WEIGHT, WEIGHT_BOLD);
```

```
text.addAttribute(  
    TextAttribute.UNDERLINE, UNDERLINE_ON, 1, 1);
```

...

```
g2.drawString(text.getIterator(), 10, 10);
```

→ **Pokus**

ATRIBUT V POMOCNÉ STRUKTUŘE

■ Např. WPF – třída FormattedText

- `var frmText = new FormattedText(text, CultureInfo.GetCultureInfo("en-us"), FlowDirection.LeftToRight, new Typeface("Verdana"), 20, Brushes.Black);`
- `frmText.SetFontSize(30, 0, 5);`
`frmText.SetFontWeight(FontWeights.Bold, 6, 11);`
`frmText.SetFontStyle(FontStyles.Italic, 17, 5);`
`frmText.SetForegroundBrush(new LinearGradientBrush(Colors.Orange, Colors.Teal, 90.0), 6, 11);`
- `g.DrawText(frmText, new Point(10, 0));`

Lorem ipsum dolor sit amet

ATRIBUT V POMOCNÉ STRUKTUŘE

- Zásobníková struktura
 - Atribut reprezentován třídou
 - Celý řetězec rozsekán na části se stejnými atributy
- Např. C# – WPF

```
this.txtBlock.Inlines.Add(  
    new Bold(new Run("Some bold text in the paragraph.")));  
  
this.txtBlock.Inlines.Add(  
    new Run(" Some text that is not bold.")  
    { Foreground=Brushes.Red});  
  
this.txtBlock.Inlines.Add(  
    new Italic(new Bold(  
        new Underline(new Run(" And BIU :-"))))));
```



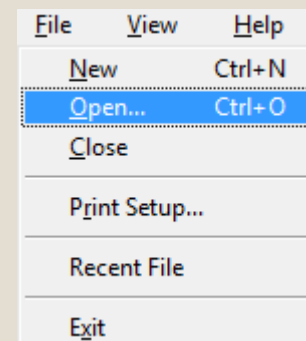
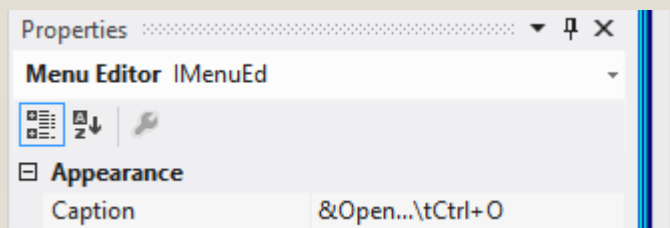
Some bold text in the paragraph. Some text that is not bold. ***And BIU :-)***

ATRIBUT V POMOCNÉ STRUKTUŘE

- Zásobníková struktura přehlednější z hlediska identifikace platnosti atributů
- Obtížnější zápis programovým kódem

ATRIBUT SOUČÁSTÍ ŘETĚZCE

- Atribut identifikován speciálním znakovým
 - Musí být definován způsob vypsání speciálního znaku
 - Tzv. escape sekvence
- Např. „&Open“ pro položku v menu
 - Escape sekvence &



ATRIBUT SOUČÁSTÍ ŘETĚZCE

- Existuje mnoho různých formátů
 - WikiText
 - Rich Text Formát (RTF)
 - Hyper Text Markup Language (HTML)
 - Extensible HyperText Markup Language (XHTML)
 - Extensible Application Markup Language (XAML)
 - Open XML Paper Specification (OpenXPS)
 - ...
- Interní implementace bývá různá

WIKITEXT

- Formát používaný na stránkách Wikipedie
- Nepříliš velký význam

`==Characteristics==`

Ward Cunningham and co-author `[[Bo Leuf]]`, in their book `''[[The Wiki Way|The Wiki Way: Quick Collaboration on the Web]]''`, described the essence of the Wiki concept as follows:

- `* A wiki ... a [[vanilla software|plain-vanilla]]`
- `* Wiki promotes ...`

Characteristics

Ward Cunningham and co-author [Bo Leuf](#), in their book [The Wiki Way: Quick Collaboration on the Web](#), described the essence of the Wiki concept as follows:

- A wiki ... a [plain-vanilla](#)
- Wiki promotes ...

RICH TEXT FORMÁT (RTF)

- Microsoft, 1987
- Určeno pro přenos atributovaného textu
 - Rozšířeno na všech platformách
- Atribut tvořen dvojicí:
 - Zpětné lomítko \
 - Klíčové slovo
 - Např. b, par, ...
- Atribut platí až do konce dokumentu
 - Výjimka: následuje atribut ukončující jeho platnost

RICH TEXT FORMÁT (RTF)

- Pro vypsání znaku označující začátek atributu se musí použít tzv. escape sekvence

- Např. \ → \\

- Příklad

- {\rtf1\ansi Text \b text \\ text.}

→

Text **text** \ **text**.

RICH TEXT FORMÁT (RTF)

- Platnost atributu lze omezit vyznačením {}
 - Atribut(y) uveden(y) v závorkách
 - Platnost skončí s uzavírací závorkou }
 - Bloky {} lze do sebe vnořovat
 - Zásobníková struktura
- Příklad
 - `{\rtf1\ansi Text {\b text} \ text.}`

→

Text **text** \ text.

RICH TEXT FORMÁT (RTF)

■ Příklad

- `{\rtf1\ansi Jedna {\b dva {\i tri} ctyri} pet.}`

→

Jedna **dva** *tri* ctyri pet.

RICH TEXT FORMÁT (RTF)

- Složitější atributy je nejprve typicky nutné nakonfigurovat na začátku textu
 - Např. používané barvy, písma, ...
- Používá se odkazem na konfiguraci
- Příklad:
 - `{\rtf1\ansi\deff0{\fonttbl{\f0 Times;}{\f1 Arial;}}{\f0 Toto je Times. }{\f1 Toto je Arial.}}`
→

Toto je Times. Toto je Arial.

RICH TEXT FORMÁT (RTF)

- Atributy lze definovat rovněž vzhled odstavce
- Příklad:

- ```
{\rtf1\ansi\deff0{\fonttbl
{\f0 Times;}{\f1 Arial;}}
{\pard\f0 Prvni odstavec.\par}
{\pard\f1\li1440 Druhy odstavec.\par}}
```

→

Prvni odstavec.

Druhy odstavec.



# RICH TEXT FORMÁT (RTF)

- Speciální atributy definují:
  - Velikost stránky
  - Počet sloupců a jejich šířku
  - Záhloví a zápatí na každé stránce
- Existují atributy pro vložení obrázku, ...

```
{\sp{\sn fLine}{\sv 0}}{\sp{\sn wzName}{\sv Obr\'e1zek
\picw17011\pich13069\picwgoal9644\pichgoal7409\pngblip
2b0e1b0000e23e49444154785eec7d07785bc7913fd17b27409060
719525ab518d54a1a8c2de7b27411044217a23f01fe6e99e205600
b45a6d4a42740ecdcf4821760881e022008f9de076887a5b0201c2
043c4780e83929a24408200410020801840042c0ef08204dec7748
a1423efafbede89d484ce0ffcbb79e7ce3bdebd134dab3579acebd
df0c218ea8eb1022802d48b0bfe1676531b005b32a5980c68219d3
d0580190e1a1412e9d5077eb2636f0873431a665e107141b8e0b68
aa22c100c3416183dac07c9b986270477bd5217842e0ce169b530f
```

# RICH TEXT FORMÁT (RTF)

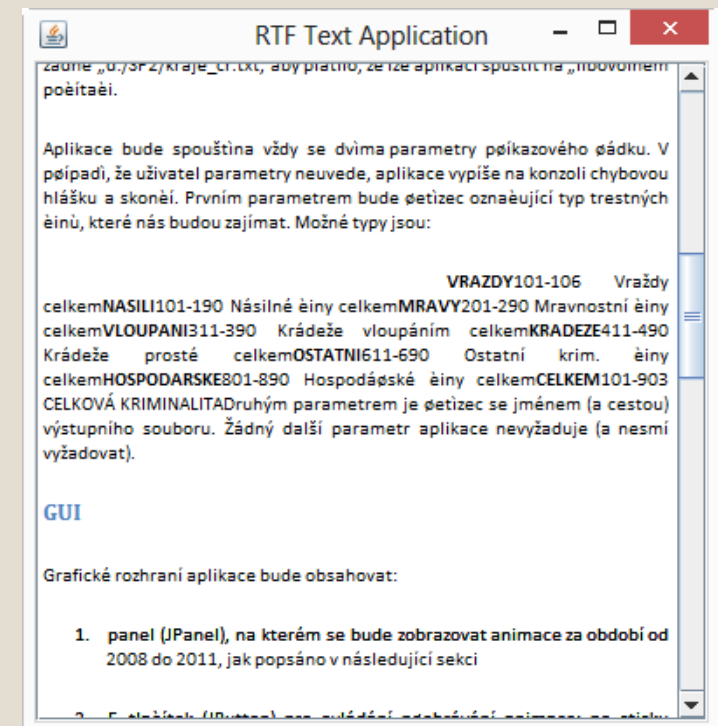
- RTF se používá často v knihovnách GUI
  - Kontrolka RichTextBox (MFC, WinForms, WPF, Java.Swing)
  - Pro rozsah textu lze nastavit barvu, font, ...
    - Nejjednodušší pro označený rozsah textu
  - Často podpora načtení/uložení z/do RTF

```
void CMFCRichTextBoxView::OnRichtexttestBold()
{
 CHARFORMAT cf;

 cf.dwMask = CFM_STRIKEOUT|CFM_BOLD;
 cf.dwEffects = CFE_BOLD;
 GetRichEditCtrl().SetSelectionCharFormat(cf);
}
```

# RICH TEXT FORMÁT (RTF)

- Implementace v knihovnách různá
  - Některé atributy mohou být ignorovány
    - Typicky tabulka, obrázek, OLE objekt



# HYPertext MARKUP LANGUAGE (HTML) / EXTENDED HTML (XHTML)

- Primárně vyvinut pro popis webových stránek
- Rozšířen na všech platformách
- Užíván často pro formátovaný text
- Atribut textu tvořen:
  - Nepárovou značkou (elementem) `<značka>`
    - Pro XHTML to musí být `<značka/>`
    - Např. `<br/>`
  - Párovou značkou (elementem) `<značka> ... </značka>`
    - Např. `<p>...</p>`
- Pro vypsání znaků `<`, `>` nutno použít escape sekvence: `&lt;` `&gt;`;

# HYPERTEXT MARKUP LANGUAGE (HTML) / EXTENDED HTML (XHTML)

## ■ Příklad

```
■ <html>
 <body>texttexttext
text</body>
</html>
```

→

text **text** text  
text

# HYPERTEXT MARKUP LANGUAGE (HTML) / EXTENDED HTML (XHTML)

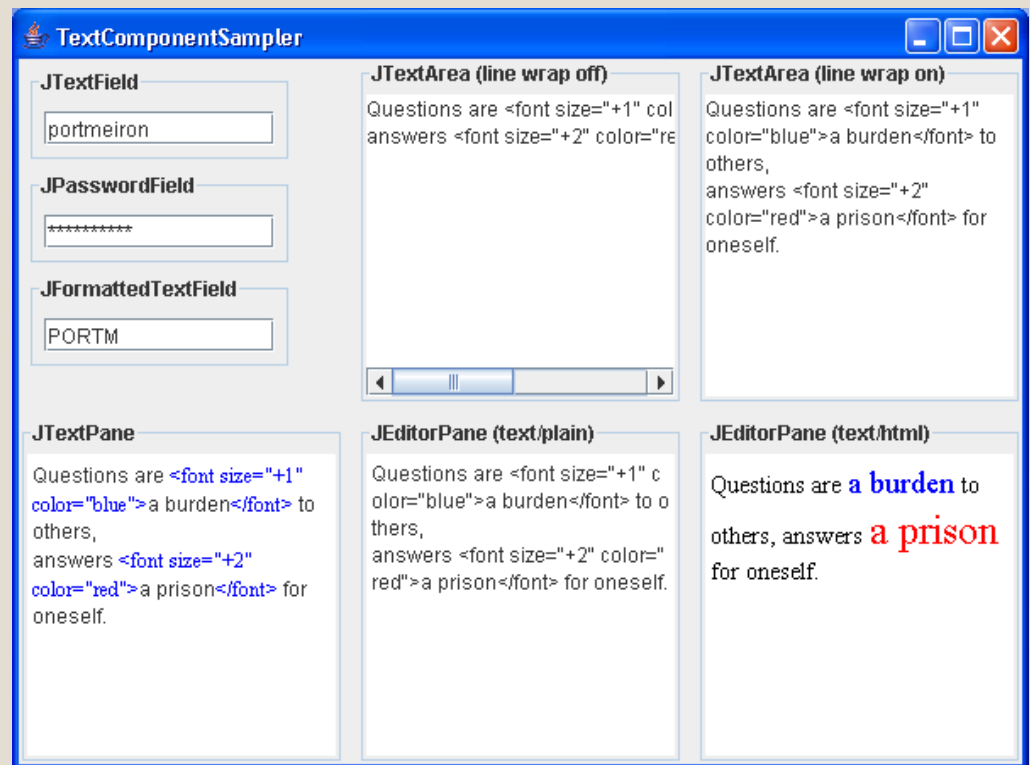
- U elementu může být upřesňující nastavení
  - Tzv. atributy, dvojice atribut="hodnota"
- Např. `<font size="12">`
- (X)HTML užívají mnohé kontrolky Java.swing
  - Příklad:

```
JButton btnn = new JButton();
btnn.setText("<html><body><font
size=\"+2\"><u>U</u>pdate</body></html>");
```



# HYPERTEXT MARKUP LANGUAGE (HTML) / EXTENDED HTML (XHTML)

- (X)HTML užívají mnohé kontrolky Java.swing
  - Příklad:



# HYPertext MARKUP LANGUAGE (HTML) / EXTENDED HTML (XHTML)

- (X)HTML obsahuje dále elementy:
  - Pro vložení bitmapového obrázku
    - ``
  - Pro vložení SVG vektorového obrázku
    - Element `<svg>`
  - Pro vložení multimediálního obsahu
    - Viz element `<video>`
    - Jen HTML 5
  - Pro odkaz jinam (na webovou stránku)
    - `<a href="stranka.html#kotva">Toto je odkaz.</a>`



# HYPertext MARKUP LANGUAGE (HTML) / EXTENDED HTML (XHTML)

## ■ Příklad:

- `<p>Trojnou uhlíkovou vazbu znázorňujeme C`  
`<svg xmlns="http://www.w3.org/2000/svg"`  
`version="1.1" width="10" height="10">`  
`<path d="M 0 1 L 10 1 M 0 4 L 10 4 M 0 7 L 10 7"`  
`style="stroke:black; fill:none;`  
`stroke-width:0.5"/></svg>C</p>`
- ➔
- Trojnou uhlíkovou vazbu znázorňujeme  $C \equiv C$

# HYPertext MARKUP LANGUAGE (HTML) / EXTENDED HTML (XHTML)

- (X)HTML umožňuje členění dokumentu do logických celků (kapitola, odstavec, ...)
  - Klíčové pro vyhledávání v dokumentech
  - Např. `<h1>...</h1>`, `<h2>...</h2>`, `<p>...</p>`, `<div>...</div>`, ...
- HTML 5 poskytuje členění dokumentu do větších logických celků (článek, sekce, ...)
  - Např. `<article>...</article>`, `<section>...</section>`,

# HYPertext MARKUP LANGUAGE (HTML) / EXTENDED HTML (XHTML)

- (X)HTML neumožňuje na rozdíl od RTF specifikovat vzhled stránky
  - Podpora stránkování zcela chybí
  - Stránka jen jedna:
    - Šířka = určena zařízením, na které se obsah zobrazuje
      - Šířka okna, šířka papíru při tisku
    - Výška neomezena
- Výhoda:
  - Obsah viditelný na libovolném zařízení bez nutnosti horizontálního posouvání
    - Text se automaticky zalomí

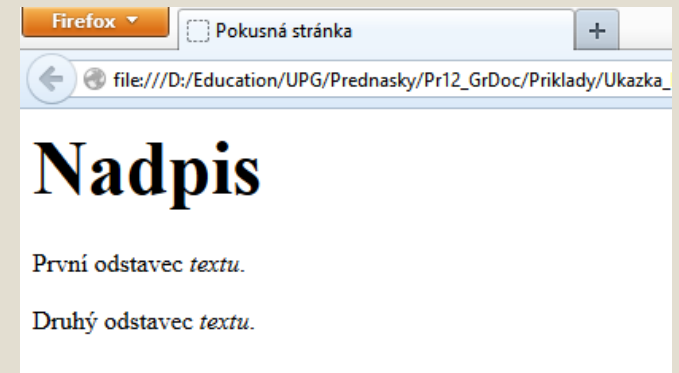
# HYPertext MARKUP LANGUAGE (HTML) / EXTENDED HTML (XHTML)

- Jen vertikální posuvník
  - Nemusí být podporováno
- Nevýhoda:
  - Autor má jen omezené možnosti definovat vzhled
  - Důsledek: při tisku může dojít k typograficky chybnému stránkování

# HYPERTEXT MARKUP LANGUAGE (HTML) / EXTENDED HTML (XHTML)

## ■ Příklad:

```
■ <html>
 <head>
 <title>Pokusná stránka</title>
 </head>
 <body>
 <!-- Logické strukturování -->
 <h1>Nadpis 1</h1>
 <p>První odstavec <i>textu</i>.</p>
 <p>Druhý odstavec <i>textu</i>.</p>
 </body>
</html>
```



# HYPertext MARKUP LANGUAGE (HTML) / EXTENDED HTML (XHTML)

- Problém: obsah a elementy pro popis logické struktury a pro popis vzhledu pohromadě
- Vzhled může záviset na koncovém zařízení
  - Např. na mobilu má být nadpis menším písmem
- Řešení: vzhled popsat styly
  - (X)HTML styly popsány CSS (Cascade Style Sheets)
- Styl může být definován:
  - Jako atribut elementu
    - Vhodné pro interaktivní manipulaci z JavaScriptu
      - Např. po najetí myší na odstavec se má zvětšit text odstavce
  - Netřeba se učit atributy vzhledu pro XHTML elementy

# HYPERTEXT MARKUP LANGUAGE (HTML) / EXTENDED HTML (XHTML)

- Předem v hlavičce dokumentu

- Např.

```
<html><head>
<title>Pokusná stránka</title>
<style>
p { text-align: left; }
h1 { text-align: left; font-size: 1.4em;
font-weight: bold; }
em { font-style: italic; }
</style>
</head>
<body>
<h1>Nadpis</h1>
<p>První odstavec textu.</p>
<p>Druhý odstavec textu.</p>
</body></html>
```

- Změnou stylů v hlavičce se změní celý vzhled dokumentu
    - Vhodné použít <em>, <strong> namísto <i> a <b>!

# HYPERTEXT MARKUP LANGUAGE (HTML) / EXTENDED HTML (XHTML)

- V externím souboru .css
  - Např. v souboru styl.css:
    - ```
p { color: #FF3030; text-align:center; }
```
 - Odkaz na soubor se vloží v hlavičce HTML dokumentu
 - `<link rel="stylesheet" type="text/css" href="styl.css">`
 - Podstrčením jiného souboru se stejným jménem lze vzhled změnit kompletně
- Přístupy lze kombinovat
 - Použije se styl, který je elementu nejbližší

HYPertext MARKUP LANGUAGE (HTML) / EXTENDED HTML (XHTML)

■ Příklad:

```
■ <html>
  <head>
    <style>
      p { color: #FF3030; text-align: center;}
    </style>
  </head>
  <body>
    <p>Červený text.</p>
    <p style="color:black;">Pokusný text</p>
    <p>Červený text.</p>
  </body>
</html>
```

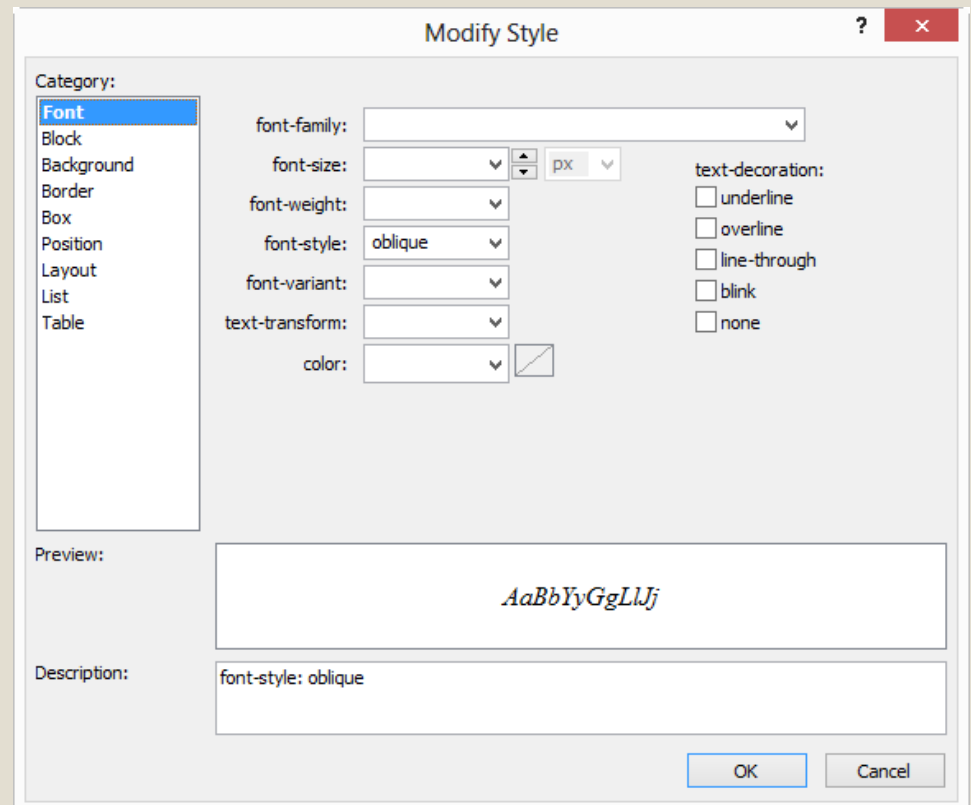
Červený text.

Pokusný text

Červený text.

HYPertext MARKUP LANGUAGE (HTML) / EXTENDED HTML (XHTML)

- Pro tvorbu stylů vhodné použít editor
 - Mnoho komerčních



HYPertext MARKUP LANGUAGE (HTML) / EXTENDED HTML (XHTML)

- Styly lze pojmenovat a pak se na ně odkazovat jejich jménem
- Příklad:

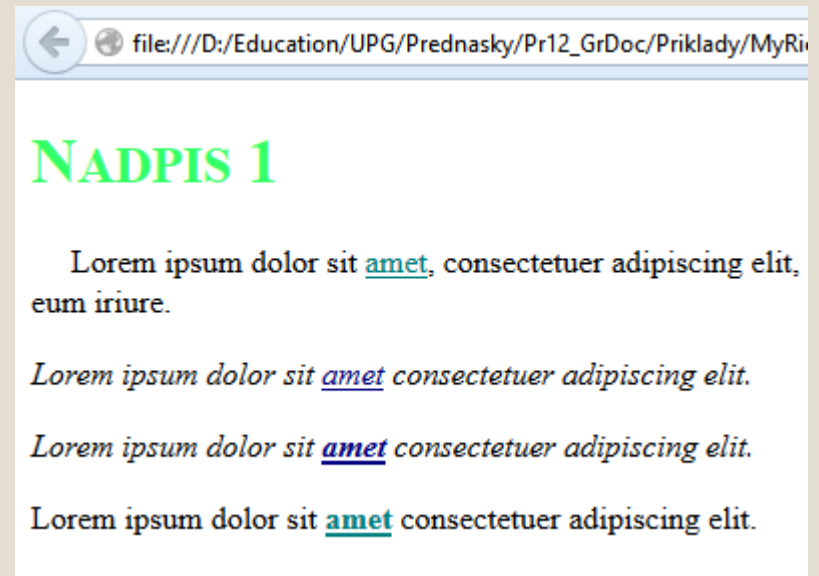
```
A:visited { color: teal; }  
A:link { color: navy; }  
A:hover { color: red; }
```

```
.prvniodstavec { text-indent: 20px; }  
.velke A { font-weight: bold; }
```

```
.zalozka { font-style: oblique; }  
.zalozka A:visited { color: navy !important; }  
H1, H2 { color: #33ff66; font-variant: small-caps; }  
H2 { font-size: 18pt; }
```

HYPERTEXT MARKUP LANGUAGE (HTML) / EXTENDED HTML (XHTML)

- `<h1>Nadpis 1</h1>`
`<p class="prvniodstavec">`
 Lorem ipsum dolor sit
 [amet](1.html) ...
`</p>`
- `<p class="zalozka">`
 Lorem ipsum dolor sit
 [amet](1.html) ...
`</p>`
- `<p class="zalozka velke">`
 Lorem ipsum dolor sit
 [amet](1.html) ...
`</p>`
- `<p class="velke">`
 Lorem ipsum dolor sit
 [amet](1.html) ...
`</p>`



HYPertext MARKUP LANGUAGE (HTML) / EXTENDED HTML (XHTML)

- Nejčastěji používané CSS atributy:
 - color – barva textu
 - background-color – barva pozadí textu
 - font-family – rodina písma
 - Serif, Sans-serif a Monospace, resp. nějaký z Web Open Font Format fontů
 - font-size – velikost písma absolutně (např. v pixelech) nebo relativně (em)
 - Preferováno je relativní
 - font-style – normal, italic, ..

HYPertext MARKUP LANGUAGE (HTML) / EXTENDED HTML (XHTML)

- font-weight – tučnost písma, např. normal, bold,
- text-align – zarovnání textu
 - left, right, center, justify
 - prohlížeč slova nedělí
- text-indent – odsazení první řádky odstavce, tzv. odstavcovou zarážku.
 - Typicky 0 (bez odsazení) nebo 1em až 3em
- vertical-align – vertikální umístěn v rámci jiného elementu
 - Např. top, bottom a middle

EXTENSIBLE APPLICATION MARKUP LANGUAGE (XAML)

- Windows Presentation Foundation (WPF)
- XAML vychází z XML
- Principy totožné s (X)HTML
 - Jiné značky pro atributovaný text
- XAML umožňuje specifikovat:
 - FlowDocument
 - FixedDocument

EXTENSIBLE APPLICATION MARKUP LANGUAGE (XAML)

■ FlowDocument:

- Obsah automaticky přizpůsobován šířce "okna"
 - Zalamování odstavců, apod.
- Vlastnosti obdobné (X)HTML
 - Určeno však pro desktopové aplikace
- WPF poskytuje třídu FlowDocumentReader pro zobrazení obsahu v aplikacích
- FlowDocument může být vložen jako obsah různých WPF kontrol (např. TextBlock, RichTextBox, ...)

EXTENSIBLE APPLICATION MARKUP LANGUAGE (XAML)

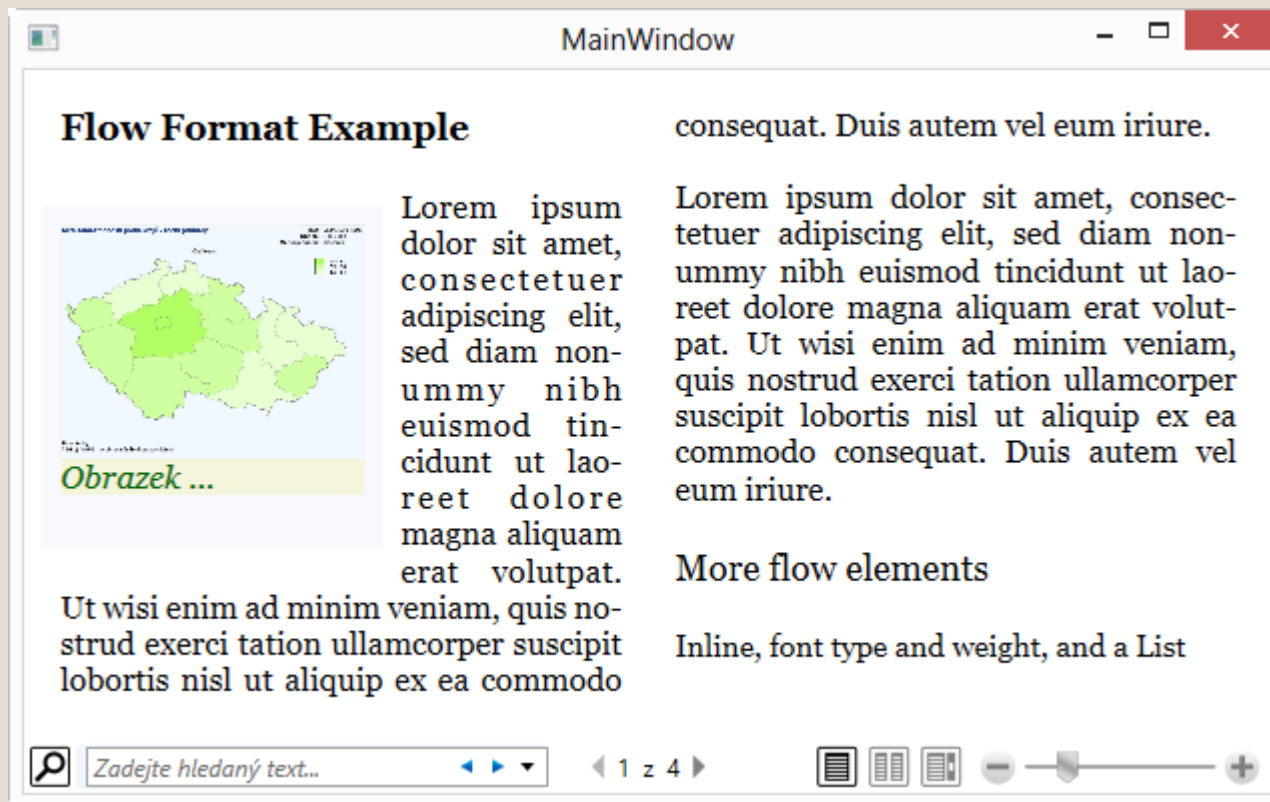
■ Příklad:

```
■ <FlowDocument ColumnWidth="6.5cm"  
  ColumnGap="0.7cm" IsHyphenationEnabled="True"  
  IsOptimalParagraphEnabled="True">  
  <Paragraph FontSize="18">  
    <Bold>Flow Format Example</Bold>  
  </Paragraph>
```

```
<Paragraph>
```

```
  Lorem ipsum dolor sit amet, consectetur ad...
```

EXTENSIBLE APPLICATION MARKUP LANGUAGE (XAML)




EXTENSIBLE APPLICATION MARKUP LANGUAGE (XAML)

- XAML podporuje styly pro jednotný vzhled

```
<TextBlock Margin="10,10,10,5" Background="DarkGray"  
           Foreground="AliceBlue" FontSize="20"  
           HorizontalAlignment="Center">
```

Toto je Text Box

```
</TextBlock>
```



```
<TextBlock>  
  <TextBlock.Style>  
    <Style>  
      <Setter Property="Margin" Value="10,10,10,5"/>  
      <Setter Property="TextBlock.Background" Value="DarkGray"/>  
      <Setter Property="TextBlock.Foreground" Value="AliceBlue"/>  
      <Setter Property="TextBlock.FontSize" Value="20"/>  
      <Setter Property="TextBlock.HorizontalAlignment" Value="Center"/>  
    </Style>  
  </TextBlock.Style>  
  Toto je Text Box  
</TextBlock>
```

EXTENSIBLE APPLICATION MARKUP LANGUAGE (XAML)

```
<Window.Resources>
    <Style TargetType="{x:Type Button}">
        <Setter Property="Padding" Value="10,5" />
    </Style>
    <Style x:Key="StatusTextStyle">
        <Setter Property="FontSize" Value="32" />
    </Style>
</Window.Resources>
```

```
<Button .../>
```

```
<TextBlock Style="{StaticResource StatusTextStyle}"/>
<TextBlock Style="{DynamicResource StatusTextStyle}"/>
```

EXTENSIBLE APPLICATION MARKUP LANGUAGE (XAML)

■ Styly lze dědit – atribut BasedOn

```
<Style x:Key="myTextStyle" TargetType="TextBlock">
    <Setter Property="Margin" Value="10,10,10,5" />
    <Setter Property="Background" Value="DarkGray" />
    <Setter Property="Foreground" Value="AliceBlue" />
    <Setter Property="FontSize" Value="20" />
    <Setter Property="HorizontalAlignment" Value="Center" />
</Style>
```

Toto je Text Box

```
<Style x:Key="myTextStyleSelected" TargetType="TextBlock" BasedOn="{StaticResource myTextStyle}">
    <Setter Property="Background">
        <Setter.Value>
            <LinearGradientBrush>
                <GradientStop Offset="0" Color="Aqua"/>
                <GradientStop Offset="1" Color="DarkBlue"/>
            </LinearGradientBrush>
        </Setter.Value>
    </Setter>
</Style>
```

Toto je Text Box

EXTENSIBLE APPLICATION MARKUP LANGUAGE (XAML)

■ FixedDocument:

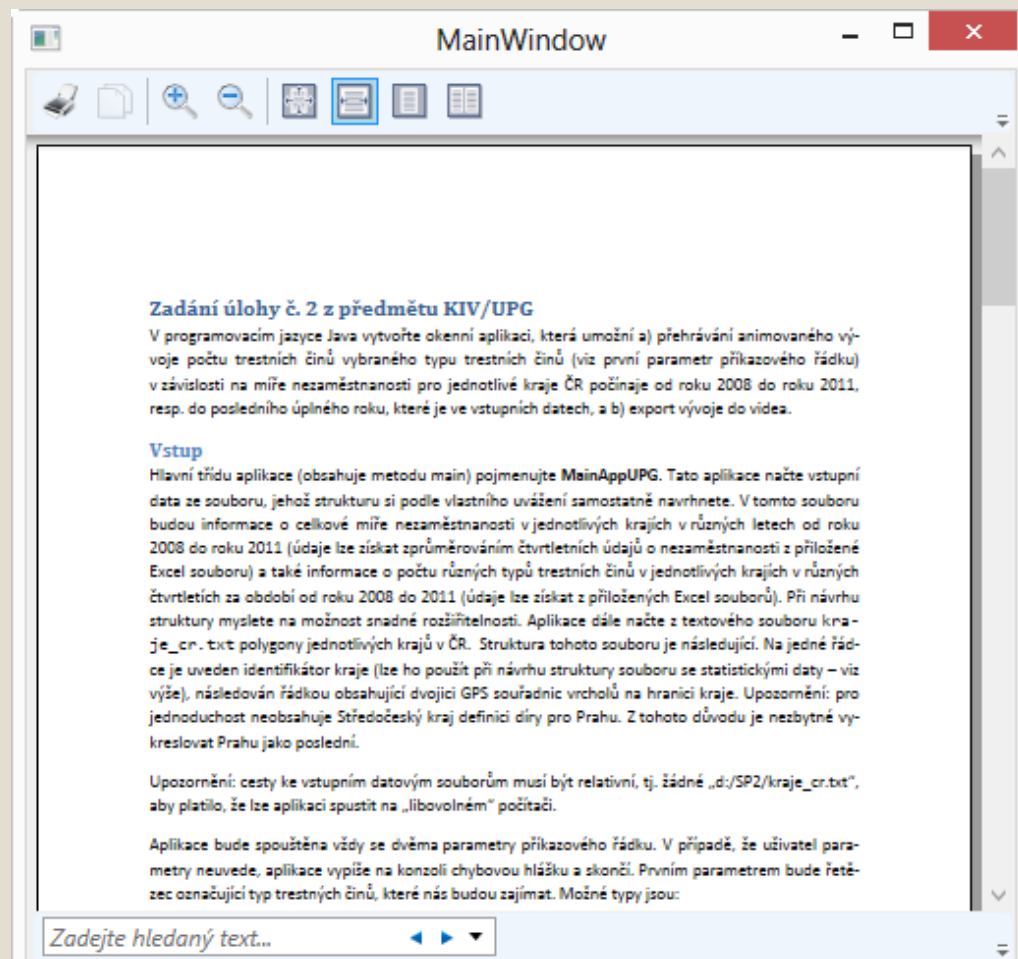
- Určeno pro WYSIWYG
- Dokument členěn explicitně na jednotlivé stránky
- Musí se explicitně říci, jak se text má zobrazit
- Příklad:

```
<FixedDocument><PageContent>  
  <FixedPage Width="21.0cm" Height="29.7cm">  
    <Canvas>  
      <Glyphs FontUri="TIMES.TTF" Fill="Black"  
        FontRenderingEmSize="14.04"  
        OriginX="70.824" OriginY="108.02"  
        UnicodeString="Jednoduchy retezec"/>  
    </Canvas>  
  </FixedPage></PageContent>
```

EXTENSIBLE APPLICATION MARKUP LANGUAGE (XAML)

- WPF poskytuje třídu `DocumentViewer` pro prohlížení
- Vytvoření "fixed" dokumentů ručně obtížné
- Řešení:
 - Programovým kódem – kreslím 2D vektorovou grafiku
 - Převodem z `FlowDocument`

EXTENSIBLE APPLICATION MARKUP LANGUAGE (XAML)



OPEN XML PAPER SPECIFICATION (OPENXPS)

- Vyvinul Microsoft
 - Postaveno na XAML technice FixedDocument
- Standardizace ECMA v roce 2006
- Určeno pro přenos dokumentů (WYSIWIG)
- Windows Vista+ obsahují XPS tiskárnu
 - Převod libovolného dokumentu do XPS
- Jedná se o soubor s příponou .xps
- Součástí OS Windows je rovněž prohlížeč XPS
- WPF poskytuje podporu prohlížení XPS

OPEN XML PAPER SPECIFICATION (OPENXPS)

- Konkurence PDF formátu
- Výhody oproti PDF:
 - XPS = ZIP archív XAML souborů → snadná editace
 - Podpora alfa kanálu v barvách
- Nevýhody:
 - Dosud nízká podpora na jiných platformách
 - Možnosti anotace jsou nižší

DALŠÍ PŘÍSTUPY

■ LaTeX

- Formátování odborného textu
- Vhodné zejména pro dokumenty s vyšším počtem matematických vzorců
 - Lze použít MathJax pro zobrazení na webu
- Viz KIV/DTP1

DALŠÍ PŘÍSTUPY

■ Příklad:

New vertex p_j such that it lies on the edge p_i, IN_2 , in a close proximity to p_i , is inserted into the triangulation and the path is altered to pass through this new vertex instead of p_i for the second time. Similarly all other edges between p_i and vertices from the chain between IN_2 and OUT_2 , excluding the edge p_i, OUT_2 , are processed. The result of the refinement is a simple contour that neither self-intersects nor self-touches -- see Fig. 7 bottom.

```
\begin{figure}
\centering
\includegraphics[width=\linewidth]{Dijkstra}
\caption{Successive searching for the shortest path passing through the given landmarks. The landmarks are denoted by small red spheres, the next pair of landmarks to be processed by larger green spheres. The path found so far is highlighted (with a self-intersection being apparent).}
\label{fig:Dijkstra}
\end{figure}
```



New vertex p_j such that it lies on the edge p_i, IN_2 , in a close proximity to p_i , is inserted into the triangulation and the path is altered to pass through this new vertex instead of p_i for the second time. Similarly all other edges between p_i and vertices from the chain between IN_2 and OUT_2 , excluding the edge p_i, OUT_2 , are processed. The result of the refinement is a simple contour that neither self-intersects nor self-touches -- see Fig. 7 bottom.



Figure 6: Successive searching for the shortest path passing through the given landmarks. The landmarks are denoted by small red spheres, the next pair of landmarks to be processed by larger green spheres. The path found so far is highlighted (with a self-intersection being apparent).

ROZLOŽENÍ GRAFICKÉHO OBSAHU

- Prvky grafického dokumentu mohou být:
 - Součástí textu
 - Volně mimo text
 - Ukotvené
- Součástí textu
 - Prvek vykreslen na místě svého uvedení
 - Standardně (X)HTML
 - Polohu lze změnit CSS
 - Standardně FlowDocument
 - Prvek vložen jako součást odstavce
 - Např. `<Paragraph>Text odstavce - začátek
<Image Source="image_0.png"/>Text</Paragraph>`

ROZLOŽENÍ GRAFICKÉHO OBSAHU

■ Volně mimo textu

- Prvek vykreslen zpravidla na místě svého uvedení
- Prvek vykreslen ve své přirozené velikosti
- Prvek může být stránkován
 - Pokud je to možné
- Prvek může být obtékán textem
 - Zleva, zprava, z obou stran

ROZLOŽENÍ GRAFICKÉHO OBSAHU

- Element Floater ve FlowDocument

- Např.

- <Paragraph>Text odstavce - začátek

- <Floater HorizontalAlignment="Center">

- <Paragraph>OBTEKANY TEXT</Paragraph>

- </Floater>Text...

- </Paragraph>

```
enim ad minim veniam, quis nostrud  
exerci tation  
ullam- OBTEKANY TEXT corper  
suscipit lobortis  
nisl ut aliquip ex ea commodo consequat.
```

ROZLOŽENÍ GRAFICKÉHO OBSAHU

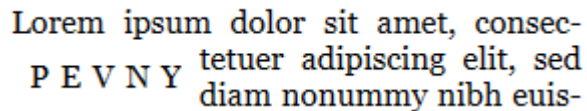
■ Ukotvené prvky

- Prvek vykreslen na pevně daném místě (X, Y)
 - Může postačovat specifikovat jen X nebo Y
- X, Y specifikováno typicky relativně vůči stránce, sloupci, obsahu, odstavci, ...
- Typicky stanovena velikost prvku
 - Obsah prvku uzpůsoben velikosti
 - Někdy relativně vůči něčemu
 - Např. při polohování GUI kontrolky lze kotvit k levému a pravému okraji zároveň → roztažení kontrolky při změně velikosti rodiče

ROZLOŽENÍ GRAFICKÉHO OBSAHU

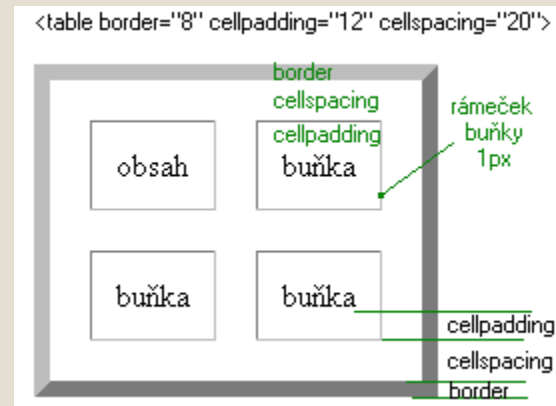
- Příklad:

- `<Paragraph>Lorem ipsum ...`
`<Figure`
`HorizontalAnchor="ColumnLeft" HorizontalOffset="0"`
`VerticalAnchor="ContentCenter" VerticalOffset="-2cm"`
`Width="2cm" Height="0.5cm" Padding="0">`
`<Paragraph>PEVNY TEXT</Paragraph>`
`</Figure>enim ad mini ...`

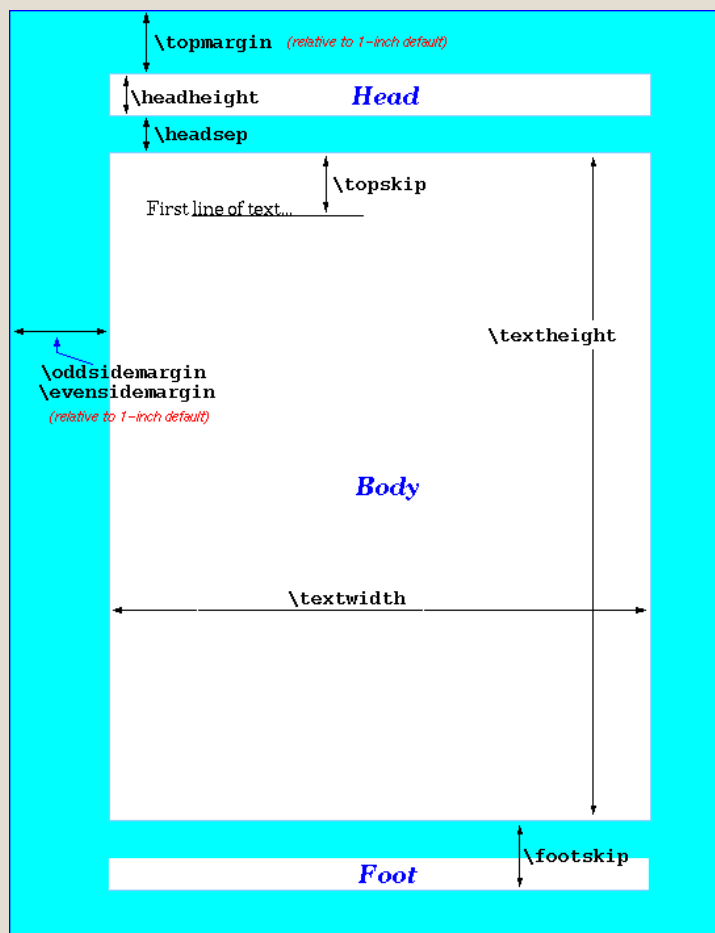


Lorem ipsum dolor sit amet, consec-
P E V N Y tetuer adipiscing elit, sed
diam nonummy nibh euismod ut imperdiet dapibus

ROZLOŽENÍ GRAFICKÉHO OBSAHU



ROZLOŽENÍ GRAFICKÉHO OBSAHU



KONEC

- Příště: zkouška 😊