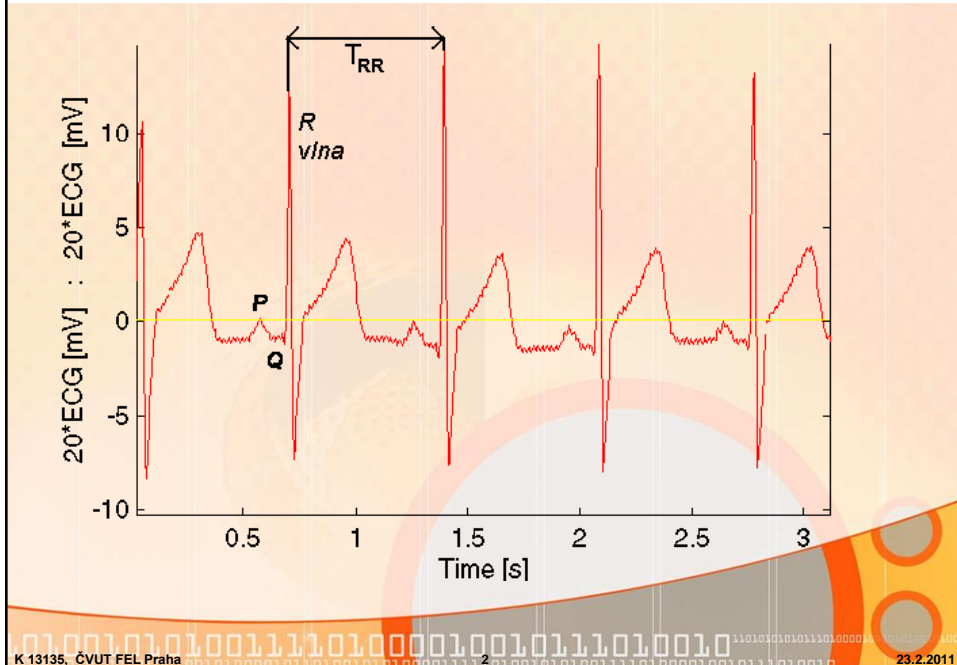


Vývoj aplikací v prostředí .NET

2. cvičení

Trochu teorie EKG



Uložení více signálů v jednom souboru

Vzorky jsou uloženy střídavě:

Příklad uložení 2 signálů (A,B) v jednom záznamu:

A[0]	B[0]	A[1]	B[1]	A[2]	B[2]
------	------	------	------	------	------

Příklad uložení 3 signálů (A,B,C) v jednom záznamu:

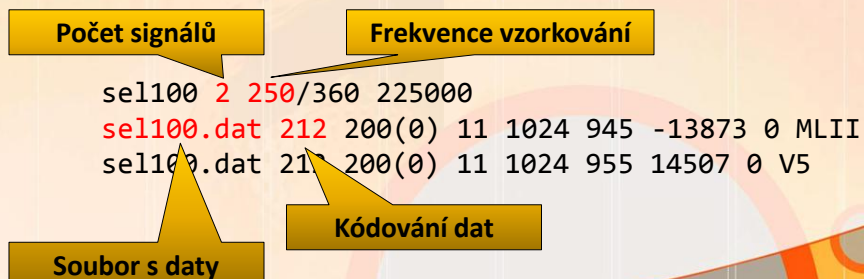
A[0]	B[0]	C[0]	A[1]	B[1]	C[1]
------	------	------	------	------	------

Hlavičkový soubor

<http://dcenet.felk.cvut.cz/testedu/ecg/>

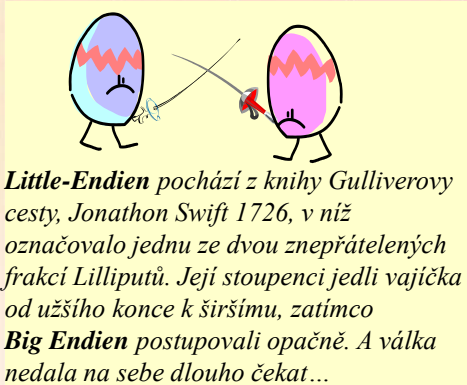
- hlavičkové soubory: *.hea
- datové soubor: *.dat

Obsah hlavičkového souboru:



Formát uložených dat

- *Naměřené vzorky se mohou ukládat jak v Little-Endian, tak Big-Endian uložení*



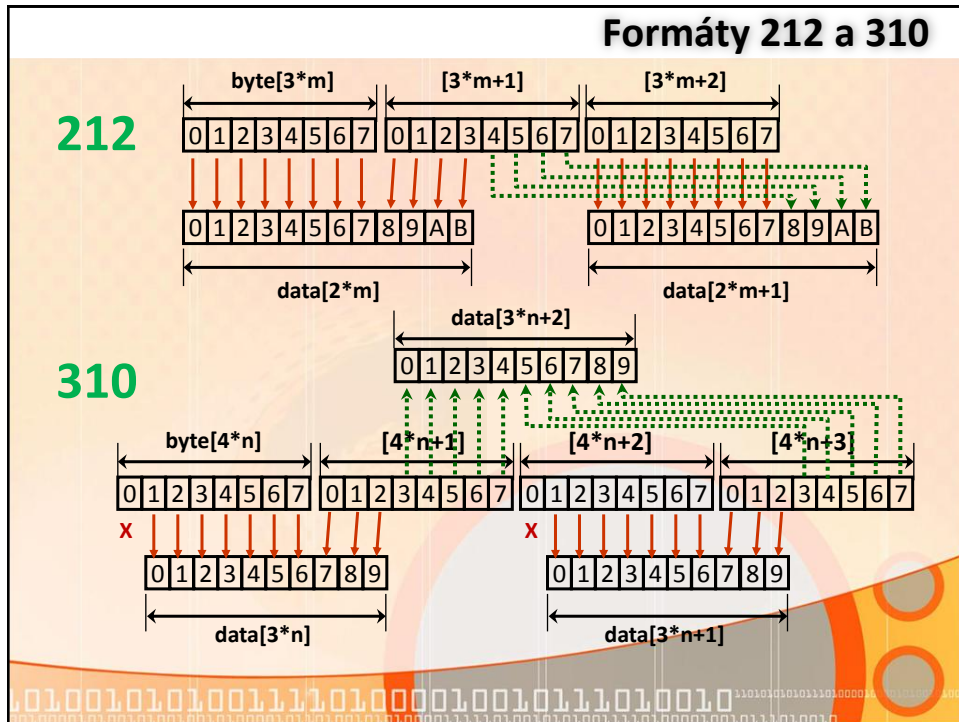
Pamatujete si, jak válka skončila?

WFDB formáty uložení měření dat

Úplný popis formátů:

<http://physionet.caregroup.harvard.edu/physiotools/wag/signal-5.htm>

- *Format 8 - první difference signálu uložené jako Int8*
- *Format 16 - Int16 uložené jako Little-Endian*
- *Format 61 - Int16 uložené jako Big-Endian*
- *Format 80 - UInt8, nula = 128*
- *Format 160 - UInt16, nula = 32768*
- **Format 212 - nejčastější** - dvě 12bitová čísla se znaménkem (tedy jakési Int12), uložená ve třech bytech po sobě jdoucích bytech, viz dále
- *Format 310 - tři 10bitová čísla se znaménkem uložená ve 4, viz dále.*
- *Format 311 - tři 10bitová čísla se znaménkem uložená ve 4 bytech, ale jinak než u 310. Mějme čtyři byty jdoucích za sebou, tedy byty s indexy $[4*n]$ až $[4*n+3]$, kde n je celé číslo skupiny. Očíslujeme v nich bity od 0 do 31. Bity 0 až 9 určují data vzorku s indexem $[3*n]$, bity 10 až 19 tvoří data $[3*n+1]$ a bity 20 až 29 dají data $[3*n+2]$. Bity 30 a 31 nejsou použité.*



Třída EcgData:

- seznam dostupných měření
- místo kde jsou dostupná měření uložena
- seznam načtených měření
- načtení dat ze souboru
- načtení hlavičkového souboru z webu
- zpracování hlavičkového souboru

Úkol 2.cvičení

EcgData
Class

Fields

- HEADER_FILE_EXTENSION : string
- list : string[]
- listRootDirectory : string
- measurements : Dictionary<string, byte[]>

Methods

- AddMeasurement() : void
- ParseHeader() : void
- ReadByteArrayFromFile() : byte[]
- ReadTextFromWeb() : string

EcgData

Seznam dostupných a načtených měření

- *Array, List, Dictionary*

Načtení dat ze souboru

```
public void AddMeasurement(  
    string dataFile, string dataEncoding,  
    int signalCount, int sampleRate)  
  
public static byte[] ReadByteArrayFromFile(string fileName)
```

Načtení a zpracování hlavičkového souboru

```
public void ParseHeader(  
    string header, out string dataEncoding,  
    out int signalCount, out int sampleRate)  
  
public static string ReadTextFromWeb(string address)
```

PRÁCE SE SOUBORY

Práce se soubory

Třídy pro práci se soubory a adresáři

System.IO.File, System.IO.Directory a System.IO.Path

Vstupy a výstupy realizovány pomocí datových proudů

zdroj → filtry → spotřebič

Práce s binárními daty

FileStream, BinaryReader, BinaryWriter

Práce s textovými daty

TextWriter, StreamWriter, StreamReader

Načtení binárních dat z lokálního souboru

```
// název souboru i s cestou (verbatim)
string jmenoSouboru = @"cesta_a_nazev_souboru";

// případné zjištění zda soubor existuje
if (File.Exists(jmenoSouboru))
{ // čtení binárních dat ze souboru
    using (FileStream fs = File.OpenRead(jmenoSouboru))
    {
        using (BinaryReader br = new BinaryReader(fs))
        {
            br.ReadInt16(); // přečte jeden Int16
        }
    }
}

// alternativní způsob - načtení celého pole
byte[] result = File.ReadAllBytes(jmenoSouboru);
}
```

Hodí se také ošetření výjimek

Vyzkoušejte snippet

- buď **Edit -> IntelliSense -> Insert Snippet Ctrl - K, X**
- nebo **try TAB TAB**

Insert Snippet:

- NetFX30
- Office Development
- Other
- Visual C#

Insert Snippet: Visual C# > t

- sim
- struct
- svm
- switch
- try**
- tryf
- unchecked
- unsafe
- using
- while

```

try
{
}
catch (Exception)
{
}

throw;
}
  
```

K 13135, ČVUT FEL
Praha

- **snippet** –malý kousek, útržek věci
(Výkladový slovník Merriam Webster:
a small part, piece, or thing;
specifically : a brief literary quotation or
quotable passage)
- **Nastavení:** Tools->Code snippet
manager
- **Vytváření:**
Help->IntelliSense Code Snippets

23.2.2011

Naše vlastní výjimky

```

public class EcgException : Exception
{
    /// <summary> stejne DEBUG i RELEASE mod. </summary>
    /// <param name="messageDebug">Zprava pro udalost.</param>
    public EcgException(string messageDebug)
        : base(messageDebug) {}

    /// <summary> Ruzne DEBUG a RELEASE mod. </summary>
    /// <param name="messageDebug">Zprava pro DEBUG mode.</param>
    /// <param name="messageRelease">Zprava pro RELEASE mode.</param>
    public EcgException(string messageDebug, string messageRelease)

    #if DEBUG
        :base(messageDebug)

    #else
        :base(messageRelease)

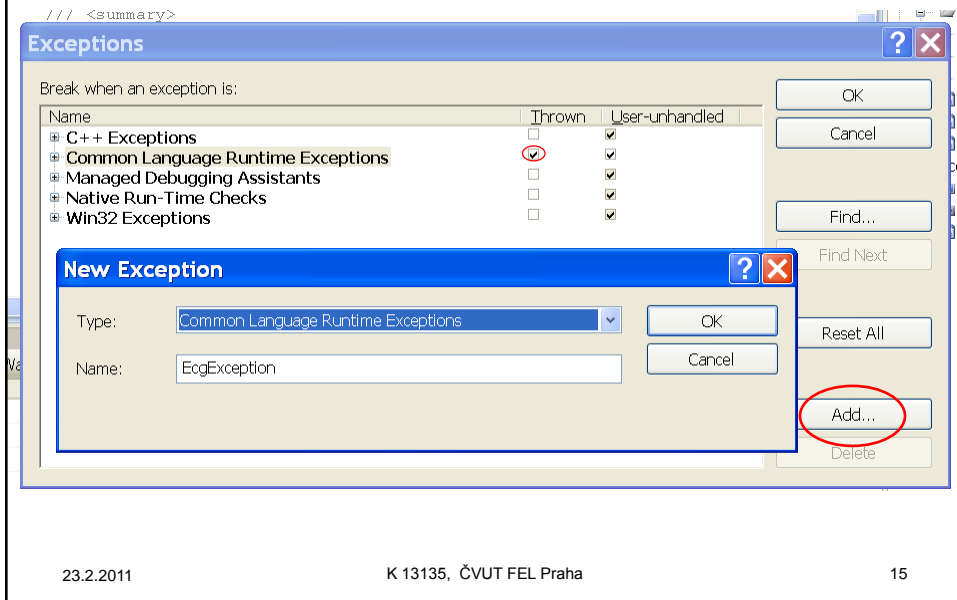
    #endif
    {}
}
  
```

23.2.2011

K 13135, ČVUT FEL Praha

14


Začleníme výjimku



REGULÁRNÍ VÝRAZY

Regulární výrazy

- Namespace *System.Text.RegularExpressions*
- C# používá pro regulární výrazy:
 - **Regex**: zadání neměnného regulárního výrazu
 - **Match**: hledá řetězce vyhovující Regex
 - **MatchCollection** : hledá kolekci vyhovujících Regex
- Zdroj <http://www.regexlib.com/CheatSheet.aspx>
- Více viz příloha k samostudiu



Seznam souborů vypíšeme takto

```

const string root = @"http://dcenet.felk.cvut.cz/testedu/ecg/";
// Vytvoříme webového klienta
System.Net.WebClient client = new System.Net.WebClient();
string page = client.DownloadString(root);
// najdeme odkazy na ecg datový soubor v hyperlincích na stránce
Regex rgx = new Regex(
    @"<a\s+href\s*=\s*[""](\w+\.\dat)[""][\.\n]*>";
// [""] značí buď dvojité uvozovky " nebo jednoduché ',
// ale " nutno v C# řetězci typu @"..." zdvojit, tedy psát jako ""
MatchCollection mc = rgx.Matches(page);
foreach (Match m in mc) Console.WriteLine(m.Value);
  
```

23.2.2011

K 13135, ČVUT FEL Praha

18

Vybraný soubor stáhneme

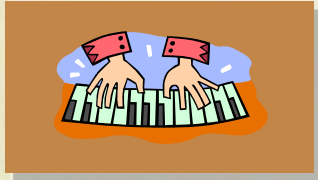
```
// vytvoříme pole všech shod - to uložíme do seznamu,
// aby se nemuselo zas vytvářet při čtení dalšího ecg souboru
MatchCollection mc = rgx.Matches(page);
Match m = mc[0]; // úplně první shoda mc[0], mc[1] další,
                // mc[mc.Count-1] poslední shoda
string ecgfile = m.Groups[1].Value; // == m.Value
// [1] závorková skupina (\w+\.\dat) ...[0] před ní, ...[2] za ní
// download datového souboru
byte[] data = client.DownloadData(root + ecgfile);
```

23.2.2011

K 13135, ČVUT FEL Praha

19

DOMÁCÍ ÚKOL NA 3. CVIČENÍ



Implementujete

- Načtení dat z webu
- Zpracování hlavičkového souboru
- Vytvoření seznamu dostupných měření
- Dekódování vstupních dat z `byte[]` na `Int16[]` obecně a specificky z kódování 212.
- Nestahujte však všechny *ecg* soubory na disk, stahuje se pouze na požádání - podívat se, zda již stažený, kdyžtak stáhnout a rozkódovat
- *Příští hodinu přezkontrolujeme, co jste s tím provedli...*

23.2.2011

K 13135, ČVUT FEL Praha

21

Připravte se na kontrolu zpracování vstupních argumentů

ZKRATKA	PLNÝ NÁZEV	POPIS
-h	--help	Nápověda
-s	--source	Soubor s naměřenými daty [URI]
-c	--encoding	Kódování dat
-n	--signalcount	Počet signálů v souboru dat
-r	--samplerate	Vzorkovací frekvence dat
-d	--header	Hlavičkový soubor [URI]
-l	--list	Seznam hlavičkových souborů [URI]
-i	--index	Index hlavičkového souboru v seznamu

```
EcgConsole -l http://dcenet.felk.cvut.cz/testedu/ecg/ -i 1
```

```
EcgConsole -s sel100.dat --encoding 212 -n 2 -r 250
```

```
EcgConsole --header C:\sel100.he
```

```
EcgConsole --help
```