Course DICis: The DICOM Medical Image Format
Chapter C1: The DICOM Header Project

Diese Übung liest unbekannte Dateien jeder Art und listet deren Inhalt, 2-stellig hexadezimal, 4-stellig dezimal und 1-stellig als Buchstaben. Falls es sich um Dicom-Bilder handelt, listet es sämtliche Felder des Dicom Headers, ohne die Bedeutung der Tags zu interpretieren. Das Programm nimmt schlicht an, dass das erste Wort und das zweite Wort der Datei die beiden Teile eines gültigen Dicoms-Tags sind und dass das dann folgende 32-Bit-Wort den Dezimalwert der Länge des ersten Eintrags enthält. Falls diese Länge kürzer als die verbleibende Dateilänge und kürzer als 1000 Byte ist, wird der Feldinhalt als String ausgegeben. Der Dateioffset wird um die Feldlänge inkrementiert und der Vorgang setzt sich fort bis zum Dateiende. Eventuell vorhandene Dicom-Pixelmatrizen sind normalerweise länger als 1000 Byte und werden deshalb nicht gedumpt.

Beispielbild: Ultrasound 303 kB
Beispielbild: Ultrasound 303 kB
Beispielbild: NMR 137 kB
Beispielbild: CT 514 kB
Beispielbild: Uro 570 kB

Projekt header1

`Microsoft Visual Studio.NET starten`
`File – New – Project – Project Types: Visual C# Projects, Templates: Windows Application`
Name: `header1`
Location: `C:\temp`
Button `OK` unten Mitte klicken.
Klicken Sie mit der **rechten** Maustaste auf des Innere von Form1.
Es öffnet sich ein kleines Kontextmenü. Klicken Sie auf View Code.
Sie sehen jetzt den vorprogrammierten Code von VisualStudio.NET. Löschen Sie den gesamten Code vollständig.

Wichtig: Bevor Sie den unten vorgegebenen Code durch kopieren nach VisualStudio.NET transferieren, müssen Sie die Formatier- und Einrückautomatik von VisualStudio.NET abschalten (sonst wird der Code von VS.NET chaotisch umformatiert):
1. Hauptmenu von Microsoft VisualStudio.NET: Klick auf Menupunkt "`Tools`".
2. Es erscheint ein DropDown-Menu. Klick auf "`Options...`".
3. Dann gehen Sie in der Unterbaum von "`Text Editor`" -> "`C#`" -> "`Tabs`". Stellen Sie `Indenting` auf `None`, `Tab size` auf `1`, `Indent size` auf `1` und schalten Sie `Insert spaces` ein.
4. Dann gehen Sie in der Unterbaum von "`Text Editor`" -> "`C#`" -> "`Formatting`". Schalten Sie alle Check Boxes von `Indentation` aus.
5. Verlassen Sie den `Options`-Dialog mit Button "`OK`".

Hexadecimal Dump

Schreiben Sie in das leere Fenster folgenden Code:

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.IO;
using System.Text;

public class Form1 : Form
{ byte[] mybuffer;//space for Dicom file
  StringBuilder s = new StringBuilder();
  TextBox TB = new TextBox();
  static void Main() {    Application.Run (new Form1( )) ;  }
  public Form1()
  { Text = "DicomHeader";
    MenuItem miOpen = new MenuItem("&Open", new EventHandler( MenuFileOpenOnClick) );
    MenuItem miExit = new MenuItem("&Exit", new EventHandler( MenuFileExitOnClick) );
    MenuItem miFile = new MenuItem("&File", new MenuItem[] { miOpen, miExit}      );
    Menu = new MainMenu( new MenuItem[] { miFile } );
    ClientSize = TB.Size = new Size( 800, 800 );
    TB.Font = new Font( "Courier New", 8 );
    TB.Multiline  = true;
    TB.WordWrap   = false;
    TB.ScrollBars = ScrollBars.Both;
    Controls.Add( TB );
    try //Delete this and the following 7 lines if you have no Internet connection.
    { System.Net.WebRequest  webreq = System.Net.WebRequest.Create(
                    "http://www.miszalok.de/C_DICis/Images/us.001" );
      System.Net.WebResponse webres = webreq.GetResponse();
      System.IO.Stream       stream = webres.GetResponseStream();
      mybuffer = new Byte[5000];
      stream.Read( mybuffer, 0, 5000 );
      DumpIt();
    } catch {};
  }

  void MenuFileOpenOnClick( object obj, EventArgs ea )
  { OpenFileDialog dlg = new OpenFileDialog();
    if ( dlg.ShowDialog() == DialogResult.OK )
    { FileStream fs = new FileStream( dlg.FileName, FileMode.Open, FileAccess.Read );
      mybuffer = new Byte[fs.Length];
      fs.Read( mybuffer, 0, (int)fs.Length );
      DumpIt();
    }
  }

  void MenuFileExitOnClick( object obj, EventArgs ea )
  { Close(); }

  void DumpIt()
  { s.Length = 0;
    int x, y;
    for ( y=0; y < 64; y++ )    // print 64 lines
    { for ( x=0; x < 16; x++ )  // 16 bytes as hexadecimals in each line
        s.Append( String.Format( "{0:X2} ", mybuffer[y*16+x] ) );
        s.Append( "\r\n" );
    }
    TB.Text = s.ToString();
    Invalidate();
  }
}
```

Übersetzen, linken und starten Sie mit `Start Without Debugging Ctrl F5`.
Öffnen Sie eine beliebige (nicht allzu große) Datei.
Sie sehen links einen 16-spaltigen Zahlenblock. In jeder Spalte stehen 2 Hexadezimalziffern, die jeweils ein Byte des Files zeigen. In jeder Zeile sehen Sie je 16 Byte des Files.

Decimal Dump

Version2: Beenden Sie Ihr Programm `header1`.
Schreiben Sie folgenden neuen Code in die Funktion `void DumpIt()` **vor** die Zeile:
`s.Append( "\r\n" );`.

```
s.Append( "  " );
for ( x=0; x < 16; x+=2 ) // 16 bytes as 8 Int16 in each line
{ int figure = mybuffer[y*16+x] + 16*mybuffer[y*16+x+1];
  s.Append( String.Format( "{0:D4} ", figure ) );
}
```

Übersetzen, linken und starten Sie mit `Start Without Debugging Ctrl F5`.
Öffnen Sie eine beliebige (nicht allzu große) Datei.
Rechts neben dem 16-spaltigen Zahlenblock steht nun ein 8-spaltiger Zahlenblock. In jeder Spalte stehen 4 Dezimalziffern, die jeweils 2 Byte des Files als Dezimalzahl zeigen.

Character Dump

Version3: Beenden Sie Ihr Programm `header1`.

Schreiben Sie folgenden neuen Code in die Funktion `void DumpIt()` **vor** die Zeile:
`s.Append( "\r\n" );`.

```
s.Append( "  " );
for ( x=0; x < 16; x++ ) // 16 bytes as 16 characters in each line
{ char c = Convert.ToChar( mybuffer[y*16+x] );
  if ( Char.IsControl(c) ) s.Append( "." ); else s.Append( c.ToString() );
}
```

Übersetzen, linken und starten Sie mit `Start Without Debugging Ctrl F5`.
Öffnen Sie eine beliebige (nicht allzu große) Datei, die Text enthält.
Rechts neben dem 16-spaltigen Zahlenblock steht ein einspaltiger Characterblock.
In jeder Zeile sind die Inhalte der drei Blöcke identisch, die drei Blöcke unterscheiden sich nur durch ihre Formatierung:
16 Spalten linker Zahlenblock: Jede Spalte enthält ein Byte = 2 Hexadezimalziffern
8 Spalten mittlerer Zahlenblock: Jede Spalte enthält ein Short Unsigned Integer = 4 Dezimalziffern.
1 Spalte rechter Characterblock: Die Spalte enthält 16 Character (nicht druckbare Zeichen als Punkt).

Dicom Dump

Version4: Beenden Sie Ihr Programm `header1`.

Schreiben Sie folgenden neuen Code in die Funktion `void DumpIt()` unter die `for`-Klammer aber noch vor der Zeile `TB.Text = s.ToString();`.

```
s.Append( "\r\n" );
int bytecount = 0;
do
{ int tlen1 = (int)mybuffer[bytecount++];
  int tlen2 = (int)mybuffer[bytecount++];
  int tlen3 = (int)mybuffer[bytecount++];
  int tlen4 = (int)mybuffer[bytecount++];
  String tag = String.Format ("({0:X4},{1:X4}) ",tlen1 + 256*tlen2,tlen3 + 256*tlen4 );
  tlen1 = (int)mybuffer[bytecount++];
  tlen2 = (int)mybuffer[bytecount++];
  tlen3 = (int)mybuffer[bytecount++];
  tlen4 = (int)mybuffer[bytecount++];
  int tlen  = tlen1 + 256*tlen2 + 256*256*tlen3 + 256*256*256*tlen4;
  s.Append( tag + String.Format("{0:D8}  ", tlen ) + "  " );
  if ( tag == "(7FE0,0010) " || tlen > 1000 ) //pixel data or too long to dump?
  { bytecount += tlen; continue; }            //do not print
  for ( int i=0; i < tlen; i++ )              //dump as characters
  { char c = Convert.ToChar( mybuffer[bytecount++] );
    if ( Char.IsControl( c ) ) s.Append( '.' ); else s.Append( c.ToString() );
  }
  s.Append( "\r\n" );
} while ( bytecount < mybuffer.Length );
```

Übersetzen, linken und starten Sie mit `Start Without Debugging Ctrl F5`. Öffnen Sie eine Dicom-Datei. Scrollen Sie in der Textbox, die `Form1` ausfüllt nach unten. Sie sehen unterhalb des Hex+Dec+Char-Dumps nun in jeder Zeile einen DICOM-Tag in folgender Form:
(1) (xxxx,xxxx) = Tag bestehend aus zwei 4-stelligen Hexadezimalzahlen laut DICOM-Spezifikation
(2) xxxxxxxx = 32 Bit Integer Zahl, welche die Anzahl der Bytes angibt, die nun folgen
(3) Inhalt des Tags als Character Dump. Nichtdruckbare Zeichen (sind meistens Teile binärer Zahlen) werden als Punkt ausgegeben.
(4) Der letzte Tag ist fast immer `(7FE0,0010)` mit großer Länge. Im File folgen hier die Grauwerte der einzelnen Pixel (nicht als Character druckbar).
Scrollen Sie in der Textbox, die `Form1` ausfüllt nach rechts. Sie sehen die Enden der langen Zeilen.

Weitere Aufgaben

(1) Stellen Sie am Bildschirm den Code der letzten `do-while`-Schleife neben ein ausgeführtes `header1`-Programmfenster.
Vergleichen Sie den Code mit dem was sie in der Ausgabe sehen und erarbeiten Sie sich den Zusammenhang.
(2) Besuchen Sie den Link: http://medical.nema.org/ und lesen Sie den Anfang von http://medical.nema.org/dicom/geninfo/dicomstrategyv105/StrategyJuly0601.htm.
(3) Besuchen Sie den Link: http://www.medizin.uni-koeln.de/zde/krz/projekte/agbildverarbeitung/vortraege/dicom.html.
(4) Besuchen Sie den Link: http://www.dicomanalyser.co.uk/html/introduction.htm.
(5) Laden und speichern Sie Beispielbilder von: http://www.xray.hmc.psu.edu/physresources/dicom/sampleimg.htm.
(6) Klicken sie auf `help` in der Menüleiste von VS.NET und suchen sie Information zu den ihnen unbekannten Sprachelementen.