

Course DICis: The DICOM Medical Image Format

Chapter C2: The DICOM Tags Project

Copyright © by V. Miszalok, last update: 07-03-2004

- ✚ [Projekt tag1](#)
- ✚ [Dump and Tag Search](#)
- ✚ [Weitere Aufgaben](#)

Diese Übung liest beim Programmstart automatisch eine Textdatei `C:\temp\dicom.dic` mit dem vollständigen und aktuellen DICOM Dictionary 2001 (bis supplement 59). Jede Zeile der Datei enthält einen Eintrag mit 5 Feldern: `Tag`, `VR`, `Name`, `VM`, `Version`. Die letzten 2 Felder werden von `tags1` ignoriert und abgeschnitten.

Man kann nunmehr unbekannte Dateien jeder Art einlesen. Das Programm listet maximal 16x16 Bytes in 3 parallelen Typkonvertierungen: Bytes hexadezimal, 16-Bit-Integer und Character.

Falls es sich um Dicom-Bilder handelt, listet es sämtliche Felder des DICOM Headers. Das Programm nimmt schlicht an, dass das erste Wort und das zweite Wort der Datei die beiden Teile eines gültigen DICOM-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 ist, werden die ersten 48 Byte des Feldinhalts als String ausgegeben. Der Dateioffset wird um die Feldlänge inkrementiert und der Vorgang setzt sich fort bis zum Dateiende.

Falls es sich um gültige DICOM-Tags handelt, interpretiert das Programm deren Bedeutung an Hand des DICOM dictionary und konvertiert DICOM-Felder vom DICOM-Typ kurze Integer (US, SS) und lange Integer (UL, SL) in Strings und zeigt diese an.

Wichtig: DICOM Dictionary ASCII-Textdatei hier laden: [dicom.dic 86 kB](#) und nach [C:\temp\dicom.dic](#) kopieren !

Vorschlag: Kopieren Sie folgende Bilder in eine neue Directory `C:\temp\Images`.

Beispielbild: [Ultrasound 303 kB](#)

Beispielbild: [Ultrasound 303 kB](#)

Beispielbild: [NMR 137 kB](#)

Beispielbild: [CT 514 kB](#)

Beispielbild: [Uro 570 kB](#)

Projekt tag1

Microsoft Visual Studio.NET starten

File - New - Project - Project Types: Visual C# Projects, Templates: Windows Application

Name: tag1

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.

Dump and Tag Search

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
    String dic; //space for dictionary file
    String DicText; //Dicom dictionary additional text;
    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 { StreamReader sr = new StreamReader( "C:\\temp\\dicom.dic" );
            dic = sr.ReadToEnd();
            sr.Close(); }
        catch { MessageBox.Show( "Cannot find C:\\temp\\dicom.dic." ); }
    }

    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(); }

    private 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 hexadecimal in each line
                s.Append( String.Format( "{0:X2} ", mybuffer[y*16+x] ) );
            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 ) );
            }
            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() );
            }
            s.Append( "\r\n" );
        }
        s.Append( "\r\n" );
    }
}
```

```

int i, 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 );
    int halftag = tlen1 + 256*tlen2;
    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 ) + " " );
    //output the first 48 bytes of any arbitrary content
    for ( i=0; i < Math.Min( tlen, 48 ); i++ )
    {
        char c = Convert.ToChar( mybuffer[bytecount + i] );
        if ( Char.IsControl( c ) ) s.Append( '.' ); else s.Append( c.ToString() );
    }
    for ( ; i < 48; i++ ) s.Append( ' ' );
    String VR = FindTagInDictionary( tag ); //function: look into the dictionary
    String Value = GetValue( VR, bytecount ); //function: get US, SS, UL, SL data types
    s.Append( ' ' + VR + Value + "\r\n" );
    bytecount += tlen;
} while ( bytecount < mybuffer.Length );
TB.Text = s.ToString();
Invalidate();
}

private String FindTagInDictionary( String tag )
{
    //uneven tags are never listed in the dictionary
    if ( Convert.ToInt16( tag[4] ) % 2 != 0 ) return "?? PrivateTag";
    int n1, n2, n3, n4;
    n1 = dic.IndexOf( tag ); //find the even tag in the dictionary
    if ( n1 < 0 ) return String.Empty; //not found
    int i = 11; // jump over the tag in order to search for the following tag
    do
    {
        n2 = dic.IndexOf( '(', n1 + i++ );//leading clause ?
        while ( dic[n2+5] != ',' || dic[n2+10] != ')' );//middle comma and end clause ?
        if ( n2 < 0 ) n2 = n1 + 13; //there is no following tag, suppress the rest
        DicText = dic.Substring( n1, n2-n1 );//cut out one line from the dictionary
        DicText = DicText.Substring( 12, DicText.Length-12 );//cut off anything in front of VR
        String VR = DicText.Substring( 0, 2 );//2 bytes of the value representation
        n3 = DicText.IndexOf( '\t', 0 ); //1. tab inside the dictionary line
        n4 = DicText.IndexOf( '\t', n3+1 );//2. tab
        String Description = DicText.Substring( n3+1, n4-n3 );//Cut out between tabs
        return VR + ' ' + Description;
    }
}

private String GetValue( String VR, int bytecount )
//This function extracts values from Dicom header tags carrying decimal data of type:
//unsigned short, signed short, unsigned long, signed long.
{
    if ( VR.Length < 2 || bytecount <= 0 ) return String.Empty; //bad parameter
    if ( VR[0] == '?' ) return String.Empty; //private tag
    if ( VR[0] == 'U' && VR[1] == 'S' || VR[0] == 'S' && VR[1] == 'S' )//16-bit integers ?
    {
        int number = (int)mybuffer[bytecount ] + 256*(int)mybuffer[bytecount+1];
        return( " = " + String.Format("{0}", number ) );
    }
    if ( VR[0] == 'U' && VR[1] == 'L' || VR[0] == 'S' && VR[1] == 'L' )//32-bit integers ?
    {
        int number = (int)mybuffer[bytecount ] + 256*(int)mybuffer[bytecount+1] +
            256*256*(int)mybuffer[bytecount+2] + 256*256*256*(int)mybuffer[bytecount+3];
        return( " = " + String.Format("{0}", number ) );
    }
}
return String.Empty;
}
}

```

Übersetzen, linken und starten Sie mit Start Without Debugging Ctrl F5.

Falls eine Fehlermeldung erscheint, steht C:\temp\dicom.dic nicht an der richtigen Stelle. Holen Sie das nach und erproben Sie tags1 erneut.

Weitere Aufgaben

Besuchen Sie den Link: http://medical.nema.org/dicom/2001/01_05PU.PDF
und lesen Sie Kapitel 6.2 ab Seite 21.

Besuchen Sie den Link: http://medical.nema.org/dicom/2001/01_06PU.PDF
und lesen Sie Kapitel 3.3 ab Seite 6.

Laden Sie `C:\temp\dicom.dic` mit Hilfe von TextPad und studieren Sie den Inhalt.

Laden und speichern Sie Beispielbilder von:

<http://www.xray.hmc.psu.edu/physresources/dicom/sampleimg.htm>.

Klicken sie auf `help` in der Menüleiste von VS.NET und suchen sie Information zu den ihnen unbekanntem Sprachelementen.

Beenden sie VS.NET, starten sie den Explorer, löschen sie die gesamte Directory `c:\temp\tags1`

Erfinden und erproben sie neue Varianten des Programms (in Form von neuen Projekten `tags2`, `tags3` usw. nach obigem Muster).