

# Urządzenia Peryferyjne

*Ćwiczenie 11 – Obsługa karty muzycznej z wykorzystaniem DirectSound, API i ActiveX*

Prowadzący: Dr Inż. Jan Nikodem

Grupa: Poniedziałek tydzień parzysty godz. 10:15

Data wykonania ćwiczenia: 08.01.2018r.

Wykonali : Paweł Biel 225949

Oskar Szubert 213624

## 1. Zagadnienia teoretyczne

### a. Plik WAVE

**WAV** (ang. *wave form audio format*) – format plików dźwiękowych stworzony przez Microsoft oraz IBM. WAVE bazuje na formacie RIFF, poszerzając go o informacje o strumieniu audio, takie jak użyty kodek, częstotliwość próbkowania czy liczba kanałów.

Kanoniczny format WAVE rozpoczyna się od nagłówka RIFF, w którym są przechowywane informacje o ID, rozmiarze pliku i formacie. Następnie w formacie WAVE znajduje się część „fmt” są tam przechowywane informacje o dźwięku np. czy dany utwór jest nagrany w mono czy stereo. Ostatnia część „data” zawiera informacje o rozmiarze danych i dane dźwiękowe. Aby odczytać informacje, która nas interesuje należy odczytać odpowiednie bajty np. odczytanie informacji o formacie : **format = fileFormat.Substring(8, 4)** oznacza to, że od 8 bajta zaczynamy czytać 4 kolejne bajty .

### *The Canonical WAVE file format*

endian	File offset (bytes)	field name	Field Size (bytes)	
big	0	ChunkID	4	The "RIFF" chunk descriptor
little	4	ChunkSize	4	
big	8	Format	4	
big	12	Subchunk1 ID	4	The "fmt" sub-chunk  describes the format of the sound information in the data sub-chunk
little	16	Subchunk1 Size	4	
little	20	AudioFormat	2	
little	22	NumChannels	2	
little	24	SampleRate	4	
little	28	ByteRate	4	
little	32	BlockAlign	2	
little	34	BitsPerSample	2	
big	36	Subchunk2 ID	4	
little	40	Subchunk2 Size	4	
little	44	data		The "data" sub-chunk  Indicates the size of the sound information and contains the raw sound data

Szczegółowy opis nagłówka WAVE (wersja spolszczona)

Struktura pliku *wave* wygląda jak poniżej:

Offset (przesunięcie)	Nazwa pola	Rozmiar w bajtach	Wartość	Opis
00	ID	04	'RIFF'	Identyfikator pliku RIFF
04	Rozmiar danych	04	Długość pliku - 8	Liczba określająca długość danych w pliku w bajtach z pominięciem pierwszych 8 bajtów nagłówka
08	Format ID	04	'WAVE'	Format pliku
12	Opis ID	04	'fmt '	Początek części opisowej pliku
16	Rozmiar opisu	04	.	Rozmiar części opisowej, dla <i>fmt</i> wynosi zwykle 16
20	Format Audio	02	0001h	Rodzaj kompresji: 1 - bez kompresji, modulacja PCM.
22	Liczba kanałów	02	0x0001, 0x0002, itd.	1 - mono, 2 - stereo
24	Częstotliwość	04	8000, 44100, itd.	Częstotliwość próbkowania w Hz
28	Częstotliwość bajtów	04	.	Częstotliwość * Liczba kanałów * Rodzicielność / 8
32	Rozmiar próbki	02	.	Liczba kanałów * Rodzicielność / 8
34	Rodzicielność	02	8, 16, itd.	Rodzicielność w bitach
36	Dodatkowe parametry	x	.	Zwykle brak tego pola
36+x	Dane ID	04	'data'	Początek części z danymi
40+x	Rozmiar danych	04	.	Rozmiar bloku danych
44+x	Dane	.	.	.

## 2. Kod programu

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Media;
using System.Text;
using System.Windows.Forms;

namespace Karta_dzwiekowa
{
    public partial class Form1 : Form
    {
        private string filePath = "";
        // private SoundPlayer player;
        private bool wasPlayed, wasRecorded = false;
        private string fileRecordPath = "";

        NAudio.Wave.WaveIn sourceStream = null;
        NAudio.Wave.DirectSoundOut soundOut = null;
        NAudio.Wave.WaveFileWriter waveFileWriter = null;

        public Form1()
        {
            InitializeComponent();
        }

        private void buttonFile_Click(object sender, EventArgs e) // Wybranie pliku do
odtworzenia
        {
            OpenFileDialog file = new OpenFileDialog();
            file.Filter = "Audio files (.wav)|*.wav"; // Akceptowalne formaty plików
to WAV
            if (file.ShowDialog() == DialogResult.OK)
            {
                filePath = file.FileName;
                labelFilePath.Text = $"Wybrany plik: {filePath}";
                FillListBox();
            }
        }

        private void FillListBox() // funkcja zaczytująca nagłówek pliku WAV
        {
            if (!string.IsNullOrEmpty(filePath)) // jeśli wybrano plik
            {
                FileStream fileStream = new FileStream(filePath, FileMode.Open,
FileAccess.Read);

                BinaryReader reader = new BinaryReader(fileStream);

                // odczytanie nagłówka pliku Wave
                byte[] wave = reader.ReadBytes(24);

                fileStream.Position = 0;
            }
        }
    }
}
```

```

int chunkID = reader.ReadInt32();
int fileSize = reader.ReadInt32();
var fileFormat = Encoding.Default.GetString(wave);
string format = fileFormat.Substring(8, 4);
string subchunk1ID = fileFormat.Substring(12, 8);
int subchunk1Size = reader.ReadInt32();

reader.Close();

// Przypisanie odczytanych wartości do zmiennych typu string
string chunkIDStr = $"Chunk ID: {chunkID}";
string fileSizeStr = $"Chunk size: {fileSize}";

string fileFormatStr = $"Format: {format}";
string subchunk1IDStr = $"Subchunk ID: {subchunk1ID}";
string subchunk1SizeStr = $"Subchunk Size ID: {subchunk1Size}";

// Wyświetlenie danych w oknie
listBoxFileInfo.Items.Clear();
listBoxFileInfo.Items.AddRange(new string[]
{
    "\tNagłówek pliku:", chunkIDStr, fileSizeStr, fileFormatStr, "\tOpis
struktury audio:", subchunk1IDStr
    , subchunk1SizeStr
});
}
}

// obsługa przycisku play służącego do odtworzenia wcześniej wybranego pliku
private void buttonPlay_Click(object sender, EventArgs e)
{

    if (filePath == String.Empty)
        MessageBox.Show("Wybierz plik!"); // jeśli ścieżka do pliku jest pusta
    wyswietl komunikat o braku ścieżki
    else
    {
        SoundPlayer simpleSound = new SoundPlayer(@filePath);
        if (!wasPlayed) // jeśli nic nie jest odtwarzane
        {
            buttonPlay.Text = "STOP"; // zmien napis przycisku
            wasPlayed = !wasPlayed;
            simpleSound.Play(); // użycie funkcji play z biblioteki
SoundPlayer
        }
        else
        {
            // jeśli coś jest odtwarzane to sytuacja analogiczna tylko z
zatrzymaniem odtwarzania
            buttonPlay.Text = "PLAY";
            wasPlayed = !wasPlayed;
            simpleSound.Stop();
        }
    }
}
}

```

```

// obsługa przycisku wyszukiwania mikrofonu
private void buttonFindDevice_Click(object sender, EventArgs e)
{
    label1.Visible = true;

    List<NAudio.Wave.WaveInCapabilities> sources = new
List<NAudio.Wave.WaveInCapabilities>();

    for (int i = 0; i < NAudio.Wave.WaveIn.DeviceCount; i++)
        sources.Add(NAudio.Wave.WaveIn.GetCapabilities(i));

    listBoxDevices.Items.Clear(); // jeśli już coś było w oknie to je wyczyść

    int counter = 0;
    foreach (var source in sources)
    {
        string item = source.ProductName;
        listBoxDevices.Items.Add("Mikrofon " + counter + "->" + item); // dodanie i
wyswietlenie do okna znalezionych mikrofonów
        counter++;
    }
}

private void sourceStream_DataAvailable(object sender, NAudio.Wave.WaveInEventArgs
e)
{
    if (waveFileWriter == null)
        return;

    waveFileWriter.Write(e.Buffer, 0, e.BytesRecorded);
    waveFileWriter.Flush();
}

// obsługa przycisku nagraj
private void buttonRecord_Click(object sender, EventArgs e)
{
    // jeśli coś nie jest nagrywane
    if (wasRecorded == false)
    {
        // jeśli nie znaleziono urządzeń (mikrofonu)
        if (listBoxDevices.SelectedItems.Count == 0)
            return;

        // jeśli nie została wybrana ścieżka do zapisu nagrania wyświetl komunikat
        if (fileRecordPath == "")
        {
            MessageBox.Show("Wybierz miejsce w którym chcesz zapisać plik!");
        }
        else
        {
            // nagrywanie do wcześniej wybranego pliku
            int deviceNumber = listBoxDevices.SelectedIndex;

            sourceStream = new NAudio.Wave.WaveIn();

```

```

        sourceStream.DeviceNumber = deviceNumber;
        sourceStream.WaveFormat = new NAudio.Wave.WaveFormat(44100,
NAudio.Wave.WaveIn.GetCapabilities(deviceNumber).Channels); // nadanie czestotliwosci
nagrywania, i standardu mono czy stereo wynikajacego z urzadzenia

        sourceStream.DataAvailable += new
EventHandler<NAudio.Wave.WaveInEventArgs>(sourceStream_DataAvailable);
        waveFileWriter = new NAudio.Wave.WaveFileWriter(fileRecordPath,
sourceStream.WaveFormat);

        sourceStream.StartRecording();

        buttonRecord.Text = "Nagrywanie...";
        wasRecorded = true;
    }
}
else if (wasRecorded == true) // jesli jest juz coś nagrywane to zatrzymaj
obecne nagrywanie i zmien tekst na przyciskach
{
    if (soundOut != null)
    {
        soundOut.Stop();
        soundOut.Dispose();
        soundOut = null;
        buttonRecord.Text = "Nagraj";
    }
    if (sourceStream != null)
    {
        sourceStream.StopRecording();
        sourceStream.Dispose();
        sourceStream = null;
        buttonRecord.Text = "Nagraj";
    }
    if (waveFileWriter != null)
    {
        waveFileWriter.Dispose();
        waveFileWriter = null;
        buttonRecord.Text = "Nagraj";
    }
    labelRecording.Text = "";
}
}

// funkcja wyboru sciezki do zapisywania nagrania
private void button1_Click_1(object sender, EventArgs e)
{
    SaveFileDialog save = new SaveFileDialog();
    save.Filter = "Wave File (*.wav)|*.wav";
    if (save.ShowDialog() != DialogResult.OK) return ;
    else
        fileRecordPath = save.FileName;
}

private void Form1_Load(object sender, EventArgs e)
{
}
}

```

```
private void label1_Click(object sender, EventArgs e)
{
}

private void groupBox1_Enter(object sender, EventArgs e)
{
}
}
```