PEMBUATAN APLIKASI KOMPRESI FILE WAVE DENGAN METODE ALGORITMA HUFFMAN MENGGUNAKAN VISUAL BASIC

Muhammad Syah[1], Naniek Widyastuti[2], Muhammad Sholeh[3]

[1,2,3] Teknik Informatika, institut Sains & Teknologi AKPRIND Yogyakarta
[1]msyah20@gmail.com, [2]naniek_wid@yahoo.com, [3]muhash@akprind.ac.id

ABSTRACT

In the Windows operating system a sound file format is widely used wave format (, WAV). Wave format is widely used for purposes such as gaming and multimedia. The format of this wave is actually a kind of rough format (raw format) in which the voice signal directly recorded and quantized into digital data. In this program will be made specifically to the compression and decompression of audio files of type wave and have manifold PCM audio formats (Pulse Code Modulation) and supports a maximum of 2 pieces of the channel channel (mono and stereo). The percentage of file compression wave does not depend on the size of the file but rather rely on the frequency of the wave file. The more repetition of data contained in the file data chunk wave then the lower the compression ratio.*

Keywords: *Compression, Wave File, Visual Basic, Audio.*

INTISARI

Dalam sistem operasi Windows salah satu *file* format suara yang banyak di gunakan adalah format *wave* (*,WAV). Format *wave* sangat banyak digunakan untuk keperluan seperti *game* dan *multimedia.* Format *wave* ini sebenarnya merupakan jenis format yang kasar (*raw format*) dimana signal suara langsung direkam dan dikuantisasi menjadi data digital. Pada program ini akan dibuat kompresi dan dekompresi khusus pada *file audio* berjenis *wave* dan mempunyai *audio format* berjenis PCM (*Pulse Code Modulation*) dan mendukung jumlah kanal maksimum 2 buah kanal (*mono* dan *stereo*). Persentase kompresi *file wave* tidak bergantung pada ukuran *file* melainkan bergantung pada *frequency* dari *file wave*. Semakin banyak perulangan data yang terdapat pada bagian *chunk* data *file wave* maka rasio kompresi akan semakin rendah.

Kata kunci: Kompresi*, File Wave*, Visual Basic, *Audio*.

PENDAHULUAN

Dalam sistem operasi Windows salah satu *file* format suara yang banyak di gunakan adalah format *wave* (*,WAV). Format *wave* sangat banyak digunakan untuk keperluan seperti *game* dan *multimedia.* Format *wave* ini sebenarnya merupakan jenis format yang kasar (*raw format*) dimana signal suara langsung direkam dan dikuantisasi menjadi data digital. Sedangkan format dasar dari *file wave* ini secara *default* tidak mendukung untuk dikompresi dan biasanya dikenal dengan nama PCM (*Pulse Code Modulation*).

Ketika ingin melakukan perekaman satu lagu pada kuliatas CD *Audio* menggunakan *sampling rate* 44,1 kHz, 16 bit per *sample,* 2 kanal (*stereo*), maka total media yang diperlukan untuk penyimpanan data *audio* per detik adalah 176.400 *byte* sehingga dalam durasi 1 menit memerlukan 10.584 MB. Jika rata-rata durasi 1 lagu selama 5 menit, maka diperlukan media penyimpanan lebih dari 50 BM untuk menyimpan data lagu tersebut. Dengan perhitungan seperti itu tentunya sangat memboroskan media penyimpanan seperti *hard disk* meskipun kapasitas *hard disk* telah disediakan dalam ukuran yang besar. Masalah tersebut dapat diatasi apabila *file wave* dikompresi dengan tujuan untuk mengurangi ukuranya.

Dengan adanya masalah pemborosan media penyimpanan diatas, maka akan di buat sebuah perangkat lunak yang dapat melakukan kompresi pada *file wave* sekaligus mampu mampu

menjalankan kembali *file wave* yang sudah dikopresi. Berdasarkan latar belakang diatas maka dalam tugas akhir ini akan dibuat denagan judul "Pembuatan Aplikasi Kompresi File Wave Dengan Metode Algoritma Huffman Menggunakan Visual Basic**".**

TINJAUAN PUSTAKA

Penelitian ini disusun berdasarkan beberapa penelitian yang telah dilakukan sebelumnya. Penelitian (Wibowo, 2012) yang berjudul "Kompresi Data Menggunakan Metode Huffman". Pada penelitian tersebut menjelaskan tentang kompresi data (pemampatan data). Kompresi data merupakan suatu teknik untuk memperkecil jumlah ukuran data (hasil kompresi) dari data aslinya. Pemampatan data digunakan untuk mengurangkan jumlah bit-bit yang dihasilkan dari setiap simbol yang muncul. Adapun kelebihan dari penelitian diatas ini adalah dapat mengkompresi beberapa bentuk *file* yaitu *file teks, image dan audio*. Sedangkan kekurangan dari penelitian diatas yaitu aplikasi yang dibuat hanya bisa memperkecil ukuran data saja tetapi tidak bisa mengembalikan ukuran data yang sudah diperkecil atau hanya bisa mengkompresi data tetapi tidak bisa melakukan dekompresi data.

Penelitian yang dilakukan (Widyarti, 2012), yang bejudul "Implementasi Algoritma Huffman pada Aplikasi Audio Compressor File Wave", menjelaskan tentang format *wave* PCM (*Pulse Code Modulation*) adalah *file wave* yang tidak terkompresi, sehingga *file wave* memiliki ukuran *file* yang sangat besar. Kelebihan dari penelitian diatas ini yaitu bisa mengimplementasikan algoritma huffman pada aplikasi kompresi *file wave* yang sudah ada. Sedangkan kekurangan dari penelitian diatas yaitu tidak membuat perangkat lunak atau aplikasi kompresi *file wave* dengan menggunakan algoritma huffman.

Adapun jurnal lain yang berjudul "Analisis Perbandingan Algoritma Huffman Dengan Algoritma (Lempel-Zip-Welch) pada Kompresi Gambar Menggunakan Metode Exponensial" (Rajagukguk, 2014), menjelaskan tentang besarnya ukuran data yang harus dikirim malampaui kecepatan transmisi yang dimiliki oleh perangkat keras yang ada, sehingga masih terdapat *delay time* yang relatif besar. Selain itu media penyimpanan seperti *hard disk* mempunyai kapasitas yang terbatas, oleh karena itu untuk mengatasi masalah ini digunakanlah kompresi data. Adapun kelebihan dari penelitian diatas adalah membuat perbandingan antara algoritma huffman dengan algoritma (Lempel-Zip-Welch) pada aplikasi kompresi. Sedangkan kekurangan dari penelitian diatas yaitu tidak membuat sebuah aplikasi atau perangkat lunak yang bertujuan untuk mengkompresi data.

Algoritma Huffman

Algoritma Huffman merupakan algoritma kompresi yang tertua yang ditemukan oleh David A. Huffman pada tahun 1952. Algoritma ini digunakan untuk membuat kompresi jenis *lossye Compression* yaitu penempatan data bahwa tidak ada satu *byte* pun data yang hilang sehingga data tersebut utuh dan disimpan sesuai dengan aslinya. Sejarahnya Huffman sudah tidak dapat membuktikan kode apapun yang efisien, ketika tugasnya hampir final didapatkan ide untuk menggunakan pohon binary untuk menyelesaikan masalahnya untuk mencari kode yang efisien (Rajagukguk, 2014).

Tahapan proses kompresi algoritma Huffman :

1. Hitung banyaknya jenis karakter dan jumlah dari masing masing karakter yang terdapat dalam sebuah *file*.
2. Susun setiap jenis karakter dengan urutan jenis karakter yang jumlahnya paling sedikit ke yang jumlahnya paling banyak.
3. Buat pohon biner berdasarkan urutan karakter dari yang jumlahnya terkecil ke yang terbesar dan memberi kode untuk tiap karakter.
4. Ganti data yang ada dengan kode bit berdasarkan pohon biner.
5. Simpan jumlah bit untuk kode bit yang terbesar, jenis karakter yang diurutkan dari frekuensi keluarnya terbesar ke terkecil beserta data yang sudah berubah menjadi kode bit sebagai data hasil kompresi.

METODOLOGI PENELITIAN

Metode Pengumpulan Data

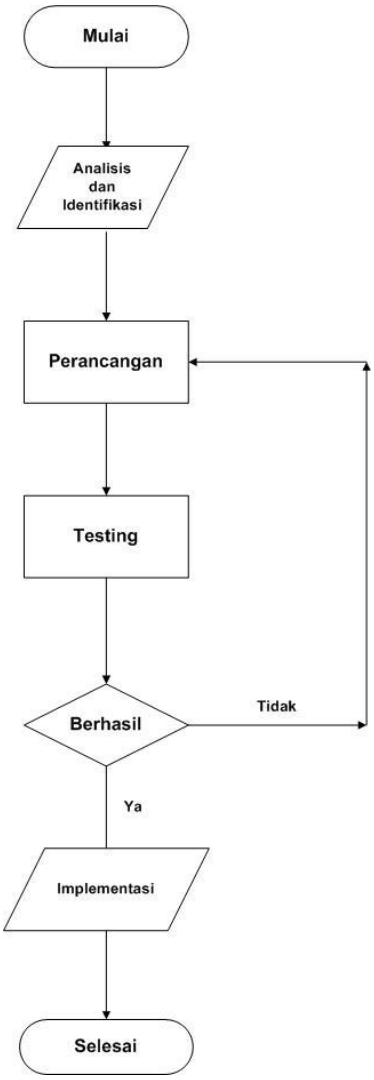
Dalam penelitian tugas akhir ini metode pengumpulan data digunakan adalah sebagai berikut :

a. Metode Studi Pustaka
   Melakukan pendalaman terhadap teori-teori yang berkaitan dengan studi kasus. Selain itu juga menggunakan beberapa jurnal yang digunakan sebagai acuan dalam menulis penelitian ini.
b. Metode Penelitian Tindakan/*Action Research*
   Dalam rangka penyelesaian penelitian ini maka digunakan metode penelitian tindakan dalam analisa, perancangan sistem, instalasi perangkat dan pengujian sistem.

Langkah dan Alir Penelitian

Dalam tahap ini, akan dibahas tentang alir penelitian yang dilakukan dari tahapan awal hingga tahapan akhir yaitu pengujian program. Berikut ini merupakan proses dari penelitian yang akan dibahas:

a. Analisis dan Identifikasi
   Pada tahap ini akan dilakukan pengumpulan data dan materi yang berguna untuk memulai rancangan dari masalah yang akan dibahas.
b. Perancangan
   Dimulainya perancangan program yang akan dibuat, termasuk didalamnya instalasi perangkat yang akan digunakan.
c. Pembuatan
   Setelah melakukkan sebuah perancangan tahap selanjutnya yaitu membuat program sesuai dengan apa yang sudah di rancang.
d. Pengujian
   Tahap pengujian ini adalah tahap dimana sebuah program yang sudah siap dan selanjutnya di uji, apabila di dalam pengujian terdapat kesalahan atau kurang sesuai dengan apa yang sudah di rancang maka diwajibkan kembali ketahap perancangan. Apabila ditahap pengujian ini program sudah berhasil maka boleh langsung ke tahap selanjutnya.
e. Implementasi
   Implementasi dilakukan setelah semua perangkat pendukung telah terinstal dan melakukan percobaan awal dari pembuatan program maka programpun telah dinyatakan selesai. Adapun bentuk diagram penelitian seperti pada gambar 1.



Gambar 1. Diagram Alir Penelitian

Alat-alat yang digunakan selama penelitian adalah sebagai berikut :

- Perangkat Keras :
  1. Memerlukan satu buah laptop atau komputer.
- Perangkat Lunak :
  1. OS Windows 7.
  2. Software Visual Basic.
  3. Corel Draw.

Perancangan

Algoritma atau *encoding* Huffman sebenarnya merupakan algoritma kompresi yang dapat diterapkan pada semua jenis baik untuk *file* biner maupun *file teks*. Algoritma ini efektif dengan rasio kompresi yang rendah jika terdapat banyak *redundancy data* atau perulangan data yang sama pada *file*.

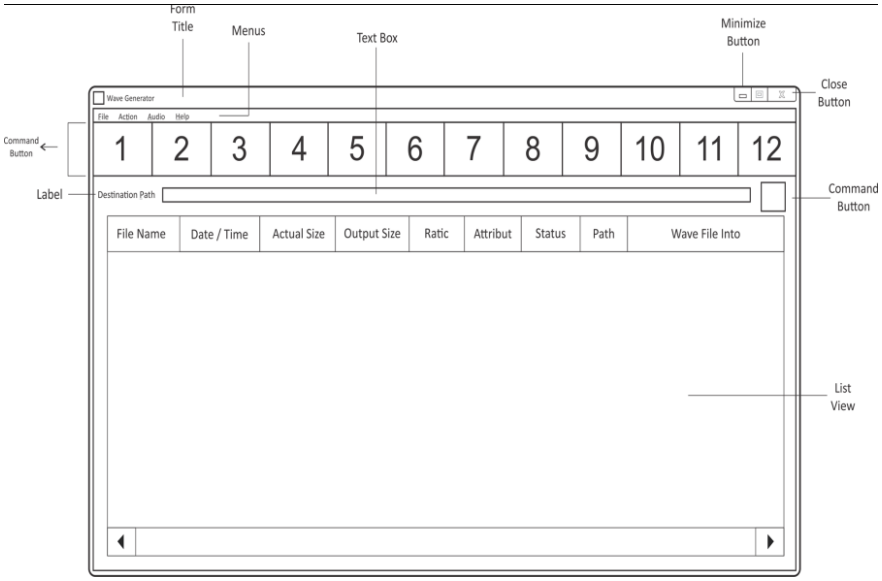
Pada program ini hanya akan dibuat kompresi dan dekompresi khusus hanya pada *file* audio berjenis *wave* dan mempunyai *audio format* berjenis PCM (*Pulse Code Modulation*) dan hanya mendukung jumlah kanal maksimum 2 buah kanal (*mono* dan *stereo*). Untuk jenis *wave* dengan *Multi Channel* tidak dapat dilakukan proses kompresi.

File wave tersebut biasanya selalu berukuran besar untuk durasi waktu main yang lama. Sebagai contoh untuk jenis *sample rate* 44.100 Hz dengan jumlah kanal *stereo* dan *bits* per *sample* 16 *bit* untuk durasi selama 1 detik saja memerlukan kapasitas sebesar $44.100 \times 2 \times 16 = 1.411.200$ *bit* per detik = 176.400 *byte* per detik. Jadi untuk durasi lagu yang rata-rata 4 menit memerlukan kapasitas $176.400 \times 4 \times 60 =$ 42.336.000 byte.

Seperti halnya dengan struktur *file* yang lain, *file wave* juga mempunyai struktur tersendiri. Struktur *file wave* mengikuti standar RIFF dengan pengelompokkan *informasi file* atas *chunk-chunk*. Secara umum bagian dari *file wave* dibagi atas bagian *header* dan bagian data. Bagian data menyimpan data *wave* yang dapat di-*playback* kembali. Sedangkan bagian *header* berisi *informasi* mengenai jenis *file wave*, *audio format*, *sample rate*, *byte rate*, jumlah kanal, *block align*, *bits per sample*, dan lain-lain.

Bagian yang dikompresi dan didekompresi pada *file wave* adalah bagian *chunk* data selain itu *file output* hasil kompresi akan diberi nilai “88” pada *sub chunk* audio *form*at untuk membedakan *file* tersebut dengan *file* tidak terkompresi yang biasanya bernilai “1” pada bagian *audio format*-nya.

Berikut ini merupakan perancangan dari *form* utama program beserta dengan komponen Visual Basic yang dipakai. Rancangan *form* utama dapat dilihat dibawah ini.



Gambar 2. Perancangan *Form* Utama

Bagian utama dari program ini dirancang dengan komponen Visual Basic seperti pada bagian atas tombol yang mempunyai *icon* biasanya disebut dengan *toolbar* tetapi pada program ini dibuat dari *command button*.

Jumlah *command button* yang berfungsi sebagai *toolbar* tersebut adalah 12 (dua belas) buah. Fungsinya dimulai dari sisi kiri ke kanan adalah sebagai berikut:

1. Tombol *command button* 1 sebagai tombol untuk menambah *file wave* tunggal ke dalam *list*.
2. Tombol *command button* 2 sebagai tombol untuk menambah semua *file wave* pada *folder* tertentu.
3. Tombol *command button* 3 sebagai tombol untuk memilih atau menandai semua *file wave* yang ada di *list*.

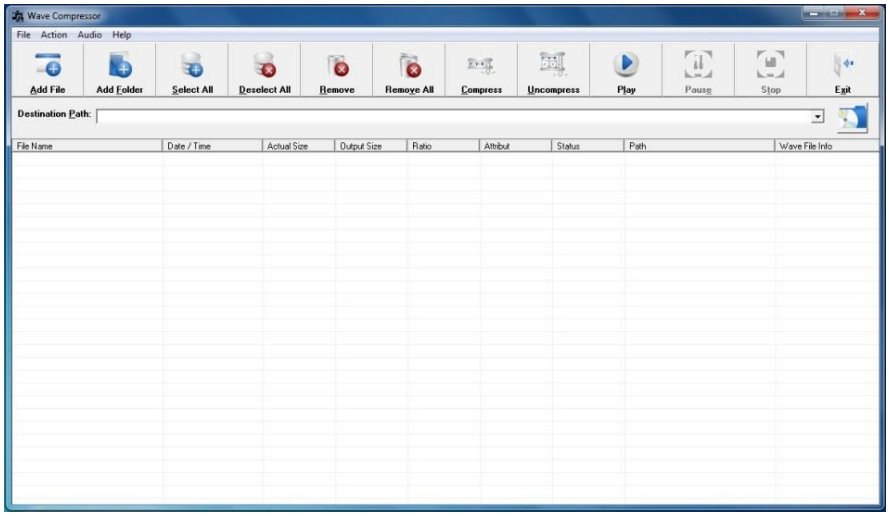
4. Tombol *command button* 4 sebagai tombol untuk menghilangkan semua tanda cek *file* pada *list*.
5. Tombol *command button* 5 sebagai tombol untuk menghapus semua *file* yang ditandai dari *list*.
6. Tombol *command button* 6 sebagai tombol untuk menghapus semua *file* baik yang ditandai atau tidak dari *list*.
7. Tombol *command button* 7 sebagai tombol untuk melakukan proses kompresi *file wave*.
8. Tombol *command button* 8 sebagai tombol untuk melakukan proses dekompresi *file wave*.
9. Tombol *command button* 9 sebagai tombol untuk memainkan *file wave* yang dipilih dari *list*.
10. Tombol *command button* 10 sebagai tombol untuk menghentikan sejenak *file wave* yang sedang dimainkan.
11. Tombol *command button* 11 sebagai tombol untuk menghentikan *file wave* yang sedang dimainkan.
12. Tombol *command button* 12 sebagai tombol untuk keluar dari program.

Bagian lainnya adalah sebuah *text box* "Destination Folder" tempat menampung *string path folder output*. Untuk memilih *folder output* tersebut dapat dengan mengklik pada tombol di samping kanan *text box* tersebut ataupun dengan cara mengetikkan secara langsung pada *text box* tersebut.

PEMBAHASAN

Tampilan Utama Program

Bentuk antarmuka program ini dibuat sesederhana mungkin, yang bertujuan agar pengguna lebih mudah untuk mengoperasikanya. Adapun bentuk program utama dapat dilihat pada gambar yang dibawah ini.



Gambar 3. Tampilan Utama Program

Pada tampilan utama program terdapat beberapa tombol yang berfungsi sesuai dengan fungsinya masing-masing, adapun nama dan fungsi tombol dimulai dari sisi kiri ke kanan adalah sebagai berikut:

1. Tombol "**Add File**" yaitu berfungsi untuk menambahkan *file wave* secara tunggal kedalam *list*.
2. Tombol "**Add Folder**" yaitu berfungsi untuk menembahkan semua *file wave* yang berada pada folder tertentu kedalam *list.*
3. Tombol "**Select All**" yaitu berfungsi untuk memilih atau menandai semua *file wave* yang berada pada *list*.
4. Tombol "**Deselect All**" yaitu berfungsi untuk menghilangkan semua tanda cek *file* pada *list*.

5. Tombol "**Remove**" yaitu berfungsi untuk menghapus semua *file* yang sudah ditandai dari *list*.
6. Tombol "**Remove All**" yaitu berfungsi untuk menghapus semua *file* yang berada di *list* baik yang ditandai maupun yang tidak ditandai.
7. Tombol "**Compress**" yaitu berfungsi untuk melakukan proses kompresi *file wave*.
8. Tombol "**Uncompress**" yaitu berfungsi untuk melakukan proses dekompresi *file wave*.
9. Tombol "**Play**" yaitu berfungsi untuk memainkan *file wave* yang dipilih dari *list*.
10. Tombol "**Pause**" yaitu berfungsi untuk menghentikan sejenak *file wave* yang sedang dimainkan.
11. Tombol "**Stop**" yaitu berfungsi untuk menghentikan *file wave* yang sedang dimainkan.
12. Tombol "**Exit**" yaitu tombol yang berfungsi untuk keluar dari program.

Bagian tombol lain yang terdapat pada program utama adalah tombol "Destination Folder" yang berada tepat dibawah tombol "Exit", yang bertujuan untuk memilih lokasi folder penyimpanan *file wave* yang akan di kompresi.

Pengujian Program

Untuk mengetahui hasil pengujian program kompresi *file wave* ini dengan algoritma kompresi huffman yang telah diimplementasikan maka dilakukan pengujian pada beberapa jenis *file wave*. *File wave* yang diuji mempunyai ukuran besar yang bervariasi dan pengujian dilakukan terhadap *file wave* yang diambil melalui internet atau di *download*, berikut terdapat tabel dari hasil pengujian kompresi *file wave*.

Tambel 1. Tabel Hasil Pengujian Proses Kompresi

No	Nama File Wave	Ukuran File (byte)	Ukuran File Output (byte)	Rasio Kompresi	Lama Proses (ms)
1	Air Horn.wav	244,268	222,169	9.05%	47
2	American Woodcock.wav	1,705,004	1,583,307	7.14%	312
3	Bonapartes Gull.wav	500,012	451,189	9.76%	78
4	Brant Geese Foraging.wav	350,252	272,319	22.25%	47
5	Canadian Geese Honk.wav	164,444	128,871	21.63%	31
6	City Ambiance.wav	1,999,124	1,766,391	11.64%	358
7	Greater Black Gull.wav	307,512	269,214	12.45%	63
8	Mallard Duck.wav	360,904	241,794	33%	47
9	Ring Billed Gull.wav	571,336	501,214	12.27%	94
10	Wetlands.wav	7,188,524	6,053,923	15.78%	1,311

Dari hasil pengujian proses kompresi didapat bahwa rasio mempunyai *range* antara 7.14% untuk nilai terendah dan 33% untuk nilai yang tertinggi. Jika dicari hasil rasio kompresi tersebut secara rata-rata adalah sebesar 15.50%.

Kecepatan kompresi memang tidak dilakukan pengujian tetapi dari beberapa pengujian yang dilakukan tingkat kecepatan baik untuk proses kompresi dan dekompresi berbanding lurus dengan ukuran *file wave*, artinya semakin besar ukuran *file wave* yang diproses maka semakin lama proses berlangsung.

Berikut ini merupakan sampel dari sebuah *file wave* (Greater black Gull.wav) berukuran 307,512 *byte* yang mempunyai informasi *header* sebagai berikut:

- *Sample Rate* : 22.050 Hz
- Jumlah Kanal (*Number of Channels*) : 2(stereo)
- *Byte Rate* : 88.200 *bytes* per *second*
- *Block Align* : 4
- *Bits* Per *Sample* : 16 *bits*
- *Length* : 3.49 *seconds*

Berikut ini adalah pengujian dalam melakukan kompresi *file wave* dimana beberapa *file wave* yang dibuat dalam bentuk ukuran *file* dan durasi yang sama tetapi jenis nada yang berbeda. Untuk lebih jelasnya dapat dilihat pada tabel dibawah ini.

Tabel 2. Tabel Pengujian Ukuran dan Durasi *File* yang Sama

No	Nama File (Wave)	Durasi File Wave (detik)	Ukuran Awal File (byte)	Ukuran File Output (byte)	Jenis Nada	Rasio kompresi
1	Air Horn	2	529,288	509.545	Keras	3.73%
2	American Woodcock	2	529,288	485,942	Lembut	8.19%
3	Bonapartes Gull	2	529,288	491,307	Keras	7.18%
4	Brant Geese Foraging	2	529,288	421,873	Lembut	20.29%
5	Canadian Geese Honk	2	529,288	431,934	Lembut	18.39%
6	City Ambiance	2	529,288	452,612	Lembut	14.49%
7	Greater Black Gull	2	529,288	471,494	Keras	10.92%
8	Mallard Duck	2	529,288	371,872	Lembut	29.74%
9	Ring Billed Gull	2	529,288	482,430	Lembut	8.85%
10	Wetlands	2	529,288	442,942	Lembut	16.31%

Untuk pengujian proses kompresi *file wave* yang mempunyai ukuran dan durasi yang sama, didapat bahwa rasio mempunyai *range* antara 3.73% untuk nilai terendah dan 29,74% untuk nilai yang tertinggi. Jika dicari hasil rasio kompresi tersebut secara rata-rata adalah sebesar 13.81%.

Dari hasil penelitian diatas menunjukkan bahwa persentase kompresi *file wave* tidak bergantung pada ukuran *file* melainkan bergantung pada *frequency* dari *file wave*. Semakin banyak perulangan data yang terdapat pada bagian *chunk* data *file wave* maka rasio kompresi akan semakin tinggi.

Berikut ini merupakan sampel dari sebuah *file wave* yang mempunyai ukuran dan durasi sama, sampel *file wave*-nya adalah (Air Horn.wav) berukuran 529.288 *byte* yang mempunyai informasi *header* sebagai berikut:

- *Sample Rate* : 44.100 Hz
- Jumlah kanal (*Number of Channels*) : 2(stereo)
- *Byte Rate* : 176.400 *bytes* per *second*
- *Block Align* : 4
- *Bits* Per *Sample* : 16 *bits*
- *Length* : 3 *seconds*

Pengujian berikutnya adalah dalam melakukan kompresi *file wave*, dimana *file wave* mempunyai ukuran file yang lebih dari 10 MB, dalam pengujian *file wave* di atas 10 MB dapat dilihat pada tabel di bawah ini.

Tabel 3. Pengujian Ukuran *file* di atas 10 MB

NO	Nama File Wave	Ukuran File (byte)	Ukuran File Output (byte)	Rasio Kompresi	Lama Proses (ms)
1	DesiJourney	11,739,172	11,391,663	2.96%	2,434
2	Medley1	15,179,560	14,273,527	5.97%	3,292
3	demicheli	18,468,976	16,023,056	13.24%	3,681
4	elysium	13,670,636	11,939,885	12.66%	2,824
5	ensemble	23,118,144	19,492,641	15.68%	4,586

Dari beberapa pengujian dalam melakukan kompresi *file wave* di atas, dapat di lihat bahwa rasio kompresi tidak ada yang mencapai lebih dari 30%, karena dalam melakukan kompresi yang di utamakan adalah kualitas *file* yang sudah dikompresi, agar saat dimainkan tidak ada perbedaan antara *file* yang sudah dikompresi dengan *file* yang belum dikompresi. Jadi hasil dari beberapa pengujian dalam melakukan kompresi *file wave* adalah *file* yang sudah dikompresi dengan *file* yang belum dikompresi mempunyai kualitas suara yang sama tetapi karena sudah dilakukan kompresi maka memiliki ukuran yang berbeda.

KESIMPULAN

Berdasarkan aplikasi kompresi *file wave* yang sudah dibuat, maka dapat diambil beberapa kesimpulan sebagai berikut:

1. Dalam pembuatan aplikasi kompresi *file wave* ini digunakan metode algoritma Huffman.
2. Aplikasi yang sudah dibuat hanya bisa melakukan kompresi dan dekompresi kepada satu format *file* saja yaitu format *wave*.
3. Tingkat keamanan *file wave* setelah dilakukan kompresi cukup terjaga, dengan kata lain *file wave* tidak mengalami kerusakan setelah proses kompresi *file wave* dilakukan.
4. Kecepatan proses tidak bergantung pada data yang diproses tetapi berbanding lurus dengan ukuran *file wave*, artinya semakin besar ukuran *file wave* yang diproses maka semakin lama waktu prosesnya.
5. *File wave* yang sudah dikompresi tersebut hanya dapat dimainkan dari program ini.
6. Ukuran dan durasi *file wave* tidak berpengaruh besar dalam melakukan proses kompresi.

DAFTAR PUSTAKA

Karmela Saturnina Mega Wea, W. S. (2010). Aplikasi Player Untuk Menjalankan File Wave Yang Terkompresi Dengan Metode Huffman. *Jurnal Informatika*.

Komputer, T. P. (2001). *Pemrograman Visual Basic 6.0 .* Yogyakarta: Wahana Komputer dan Andi Offset.

Nurasyiah. (2013). Perancangan Aplikasi Kompresi Ffile Audio Dengan Algoritma Aritmetic Coding. *Pelita Informatika Budi Darma* .

Rajagukguk, D. M. (2014). Analisis Perbandingan Algoritma Huffman Dengan Algoritma (Lempel-ZIP-Welch) pada Kompresi Gambar Menggunakan Metode Exponensial. *Pelita Informatika Budi Darma*.

Setyadi, H. (2010). *Dasar Pemrograman Visual Basic.* Portal edukasi Indonesia.

Wibowo, A. (2012). Kompresi Data Menggunakan Metode Huffman. *Seminar Nasional Teknologi Informasi & Komunikasi*.

Widyarti, Y. (2012). Implementasi Algoritma Huffman pada Aplikasi Audio Compressor File Wave. *Karya Ilmiah*.