

POLITECHNIKA POZNAŃSKA

WYDZIAŁ ELEKTRYCZNY
INFORMATYKA



TEORIA INFORMACJI I KODOWANIE

DOKUMENTACJA PROJEKTU

Autorzy:

Michał MAJKA

Nr albumu: 112679

Piotr PARYSEK

Nr albumu: 106100

Prowadzący:

dr inż. Ewa IDZIKOWSKA

24 listopada 2015

1	Wstęp	1
2	Algorytm	1
	2.1 Historia	1
	2.2 Zasada działania	1
3	Opis implementacji	2
	3.1 Kodowanie	2
	3.2 Dekodowanie	5
4	Użytkowanie programu	6
5	Testy	11
	5.1 Przedstawienie wyników:	13
6	Wnioski	15
	Literatura	15

1 Wstęp

Zadaniem projektowym była implementacja algorytmu kompresji bezstratnej *Run-Length Encoding (RLE)*. Zadanie zrealizowano w środowisku programistycznym Qt Creator 5.5.1[4] korzystając z kompilatora GCC 4.9.1[5]. Do kontroli wersji oraz plików źródłowych wykorzystano oprogramowanie Git[6], projekt hostowano w repozytorium GitHub[7]. Dokumentację wykonano w L^AT_EX[1] w programie Texmaker 4.5 [2] oraz w edytorze online: ShareLaTeX[3].

2 Algorytm

2.1 Historia

Run-Length Encodings, również znane jako **Golomb Codings**, swoje „podwaliny” powstania wiąże z pracami, XVII-wiecznego francuskiego matematyka *Blaise’a Pascal’a*, związanymi z probabilistyką[9]. Koncepcja kodowania powtarzających się znaków była używana od początków istnienia teorii informacji (Shannon 1949, Laemmel 1951), jednakże metodę oraz sposób kodowania wynalazł i opracował *Solomon Wolf Golomb*[8][9].

2.2 Zasada działania

Algorytm jest relatywnie prosty → przedstawia powtarzające się wartości jako dany znak i licznik powtórzeń. Na przykład ciąg znaków:

ppppppppuuuuuttttt.....ppppooooozzzzznnnnnannnnnn....pppppplllll

Zostaje przedstawiony w postaci ciągu:

p7u5t5.6p4o6z5n4an5.4p6l5

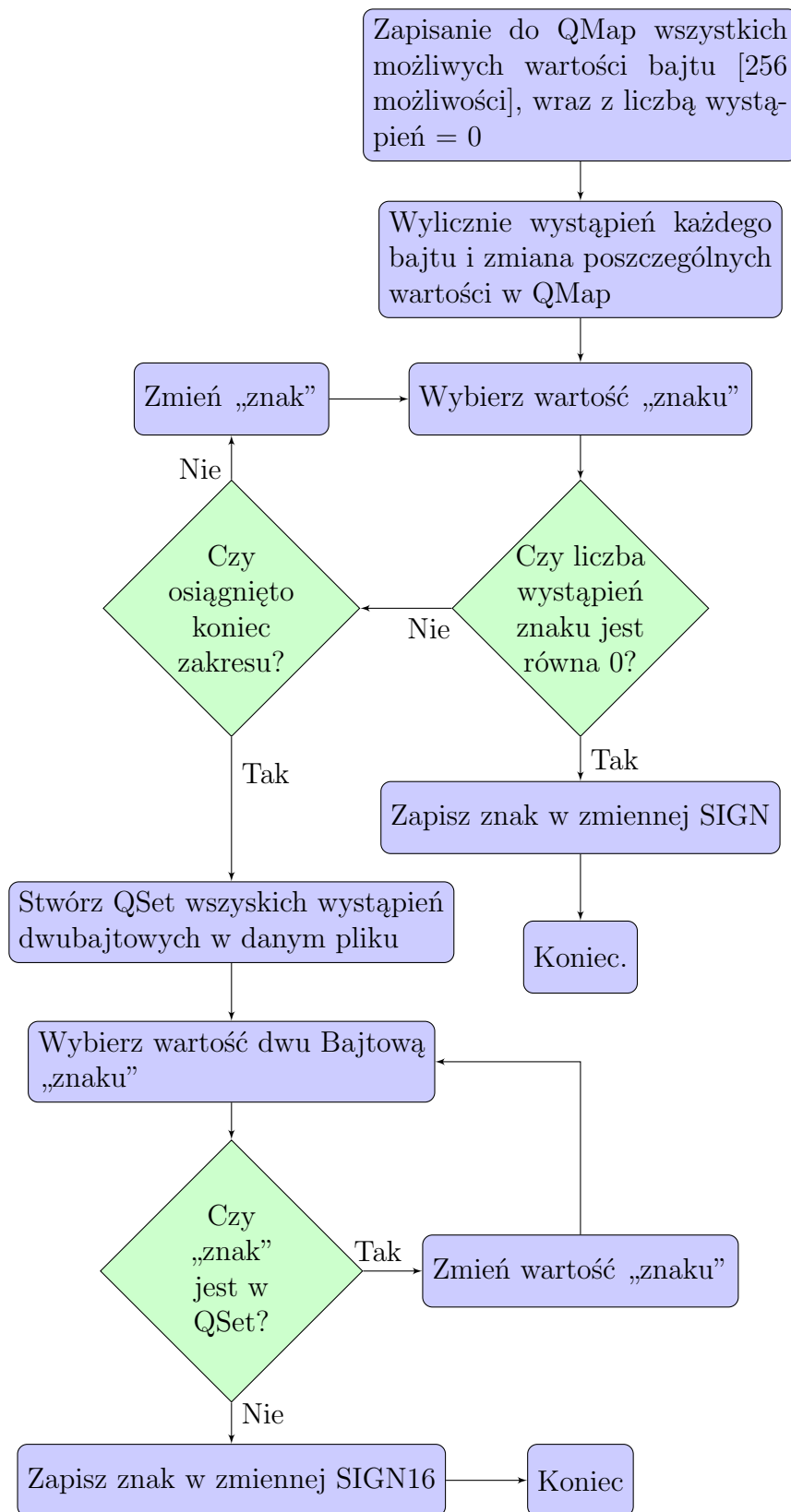
Ciąg znaków	Liczba
<i>ppppppppuuuuuttttt.....ppppooooozzzzznnnnnannnnnn....pppppplllll</i>	64
<i>p7u5t5.6p4o6z5n4an5.4p6l5</i>	26
	$26/64 \approx 0.41$

Tabela 1: Przykładowa kompresja znakowa

3 Opis implementacji

3.1 Kodowanie

Ustalenie znaku kodowania



Rysunek 1: Schemat blokowy wyszukiwania „znaku”.

Do wyszukania „znaków” wykorzystano kontener `QMap<quint8, int>`, gdzie zmienna `quint8` (unsigned byte) wskazuje na poszczególne możliwe wartości bajta, a zmienna `int` wskazuje ilość wystąpień danego bajta w opracowywanym pliku.

Kod 1: Deklaracja wraz z inicjalizacją kontenera `QMap<quint8, int>`

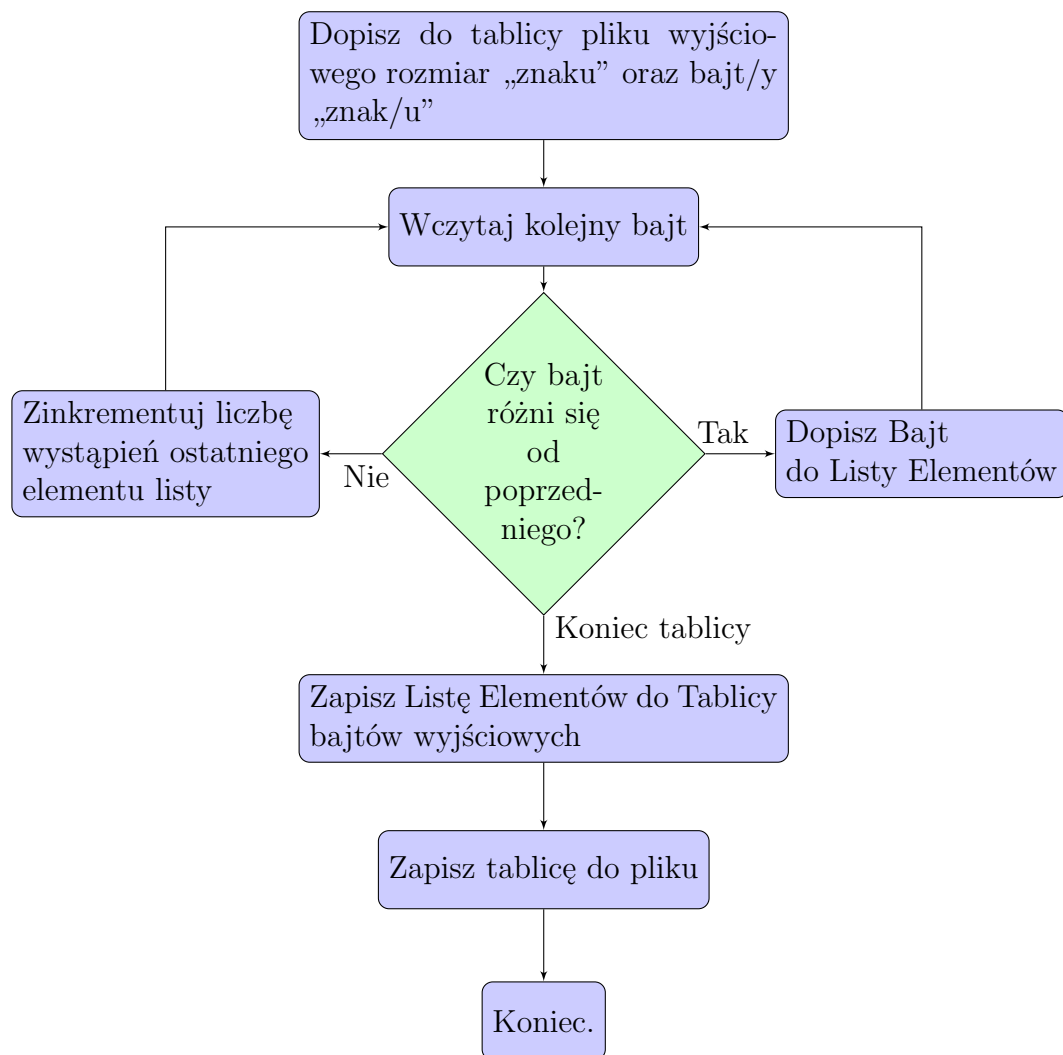
```
1 QMap<quint8, int> IIMap;
2 for (quint8 i = 0; i < 255; i++) {
3     IIMap.insert(i, 0);
4 }
```

W przypadku zdarzenia, że w pliku występują wszystkie możliwe kombinacje bajta, zamiast jedno bajtowego „znaku” zostaje wprowadzony znak dwu bajtowy:

Kod 2: Deklaracja dwu bajtowej zmiennej znakowej

```
1 QPair<quint8, quint8> SIGN16;
```

Kodowanie



Rysunek 2: Schemat blokowy kodowania pliku.

Do sprawnego wczytania, zliczenia i zakodowania poszczególnych bajtów stworzono kontener `QList` struktury `Element`. Struktura `Element` posiada dwa pola: `quint8 item` → oznaczające dany

bajt oraz quint32 value → oznaczające ilość wystąpień (Założono, że dany bajt nie powtórzy się więcej jak 4294967295 razy).

Kod 3: Główne struktury danych kodowania

```
1 struct Element {
2     quint8 item;
3     quint32 value;
4 };
5 QList<Element> Elements;
6 quint8 CurrentByte;
```

Zapisanie danych do pliku odbywa się za pośrednictwem tablicy bajtów QByteArray. Analiza zapisanych znaków odbywa się poprzez przejście przez wcześniej wspomnianą listę: QList<Element> i odpowiednią interpretację wartości wystąpień danego znaku.

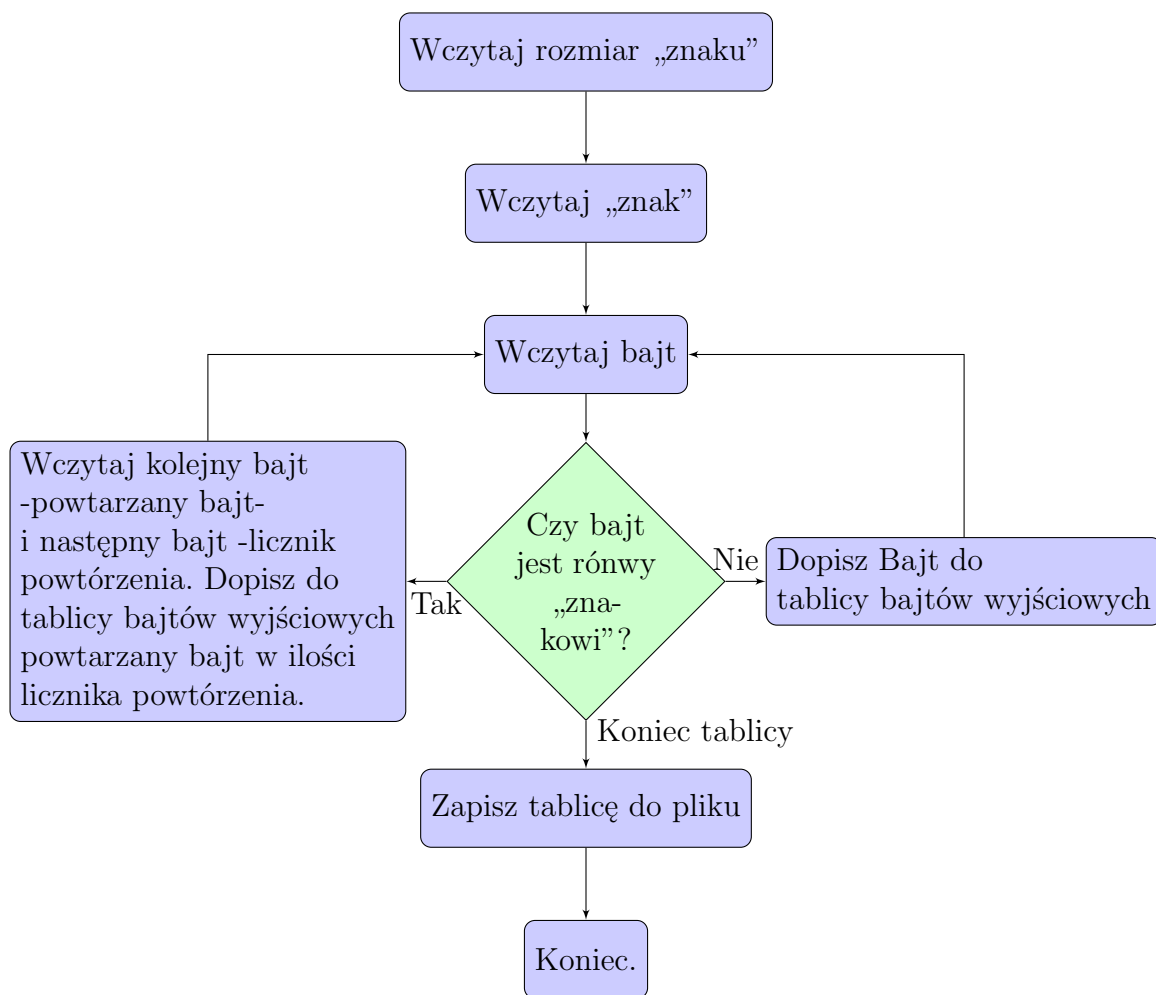
Jeżeli wartość powtórzenia danego znaku nie jest większa niż minimalna długość jego zastąpienia, który ma postać „znak” inicjujący, bajt powtarzany, ilość powtórzeń, to wpisywana jest „pierwotna postać”.

W przypadku, gdy wartość ilości powtórzeń bajtu jest większa niż maksymalna wartość jaką może osiągnąć bajt - 255 - to postać zastąpienia przybiera postać: czterech bajtów „znaku”, bajt powtarzany i cztery bajty licznika.

Kod 4: Zapisanie i zakodowania danych skompresowanych do pliku

```
1 if (e.value < 256) {
2     OutByteArray.append(SIGN);
3     OutByteArray.append(e.item);
4     OutByteArray.append(e.value);
5 } else {
6     OutByteArray.append(SIGN);
7     OutByteArray.append(SIGN);
8     OutByteArray.append(SIGN);
9     OutByteArray.append(SIGN);
10    OutByteArray.append(e.item);
11    QByteArray TempArray;
12    TempArray = RLE::IntToHex(e.value);
13    OutByteArray.append(TempArray);
14 }
```

3.2 Dekodowanie



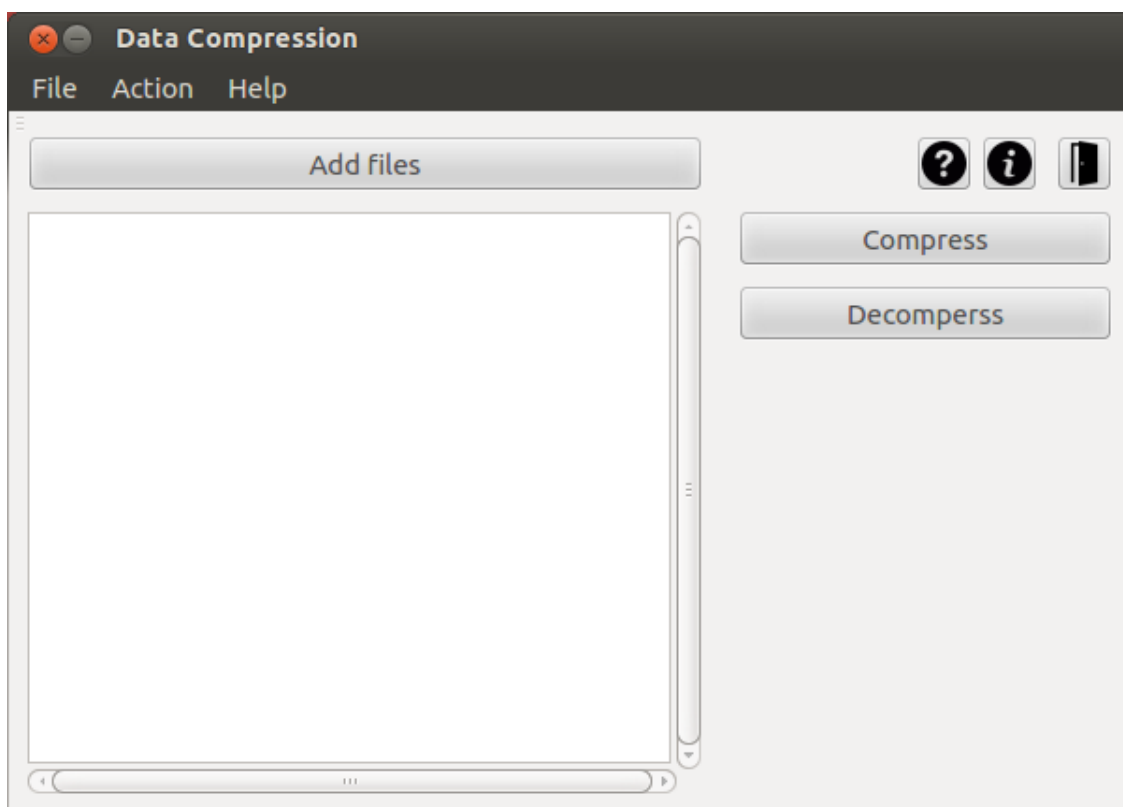
Rysunek 3: Schemat blokowy dekodowania pliku.

Dekodowanie odbywa się z pomocą analogicznych struktur / zasad / metod co zostały użyte podczas kodowania.

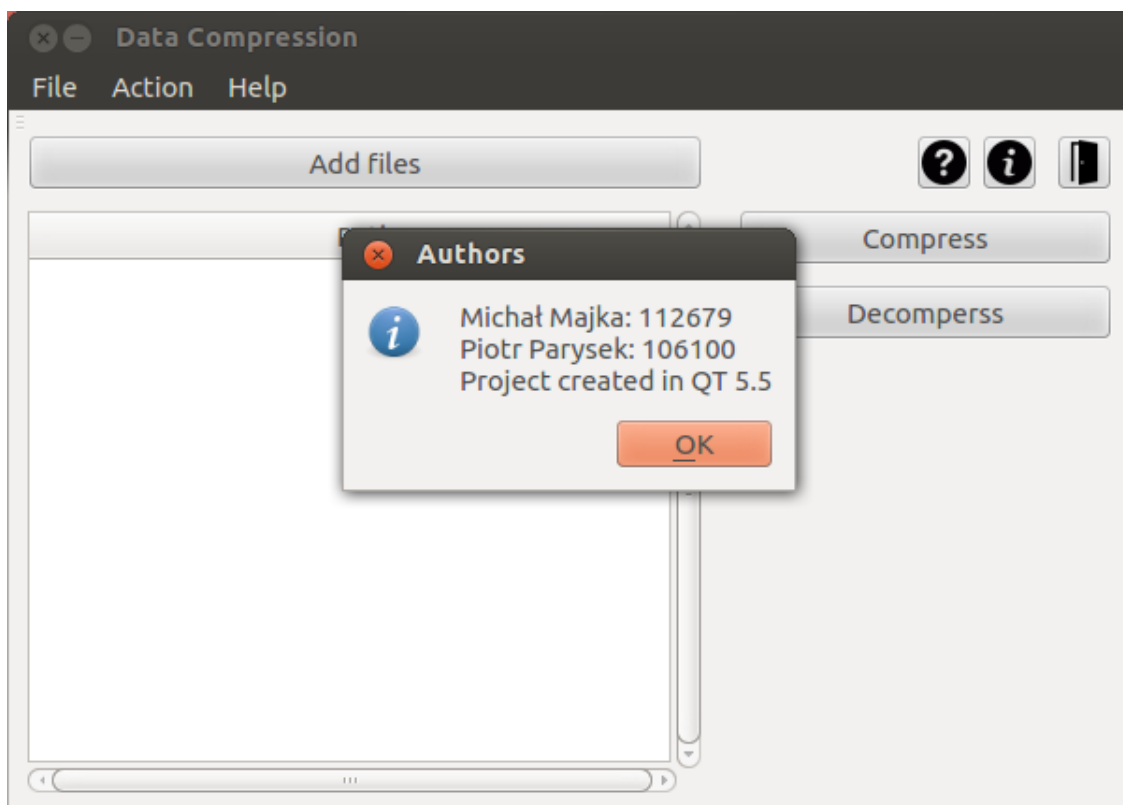
W celu ułatwienia kontroli nad plikami po kodowaniu mają one dodawany przyrostek .rlemama, a podczas dekodowania dodawany przed znacznikiem formatu pliku przyrostek _2

4 Użytkowanie programu

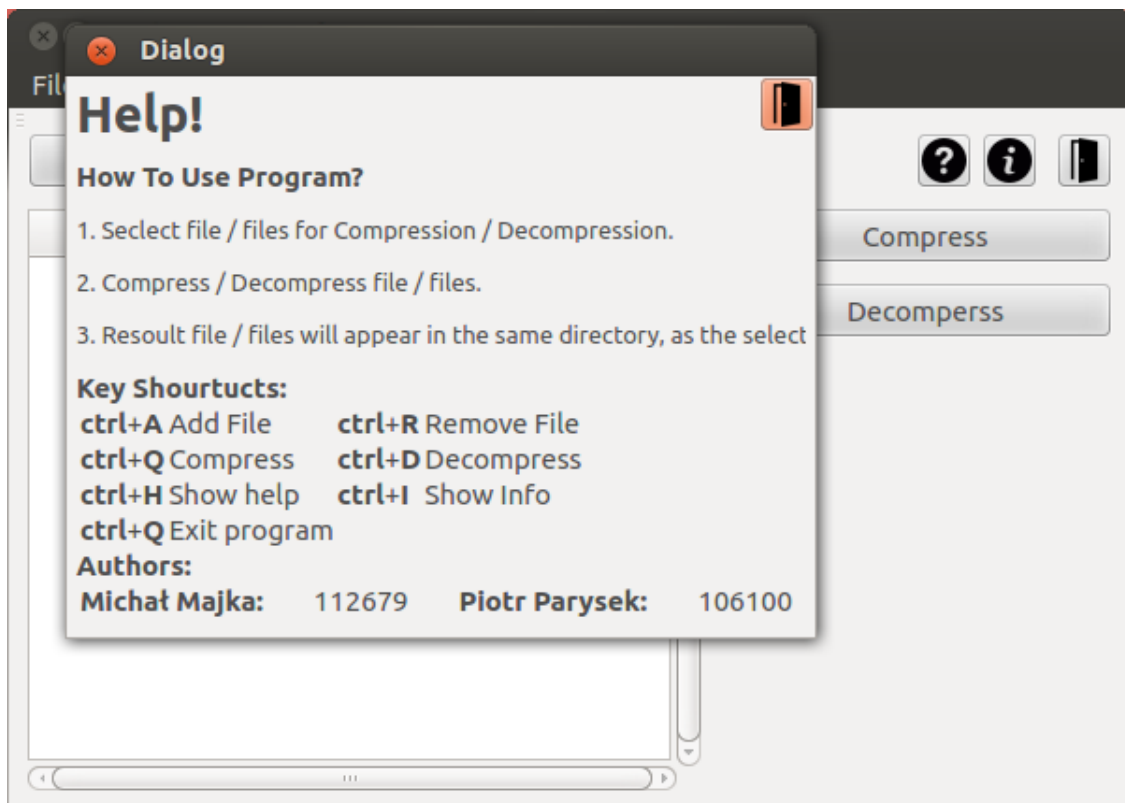
Ikony umieszczone w programie zostały pobrane z strony: <http://www.flaticon.com/>[10].



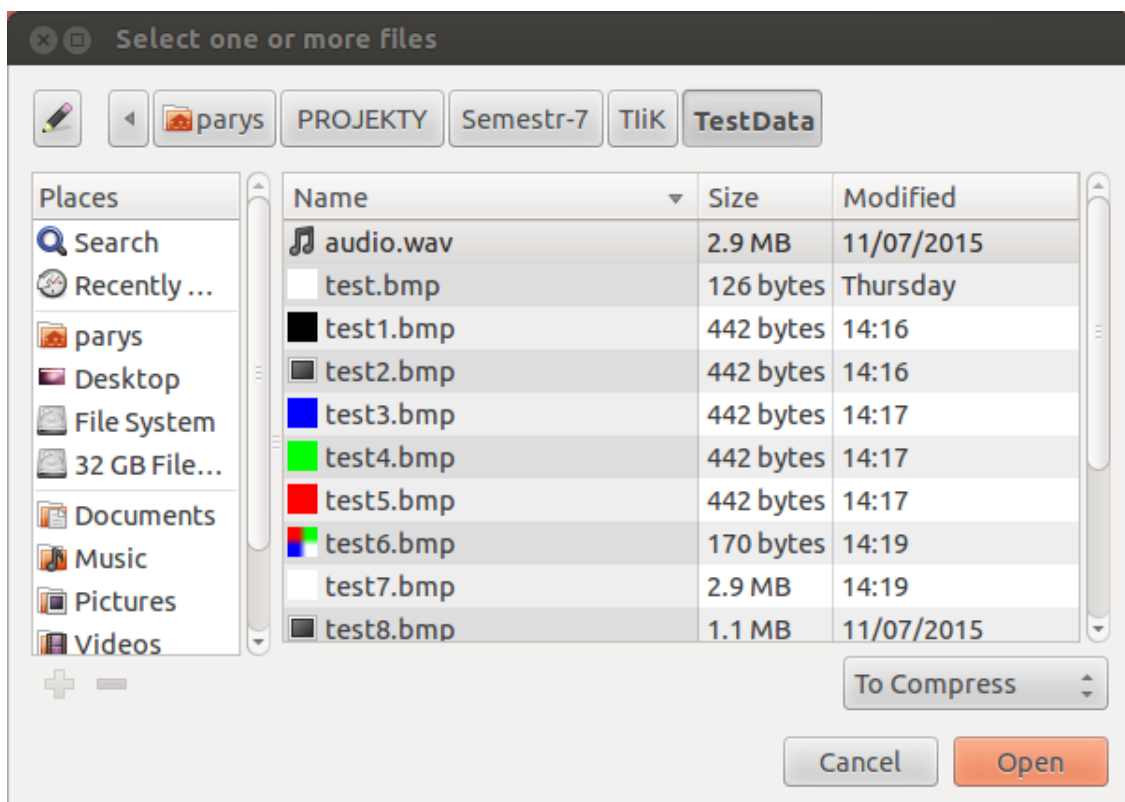
Zrzut ekranu 1: Wygląd po „starcie” programu.



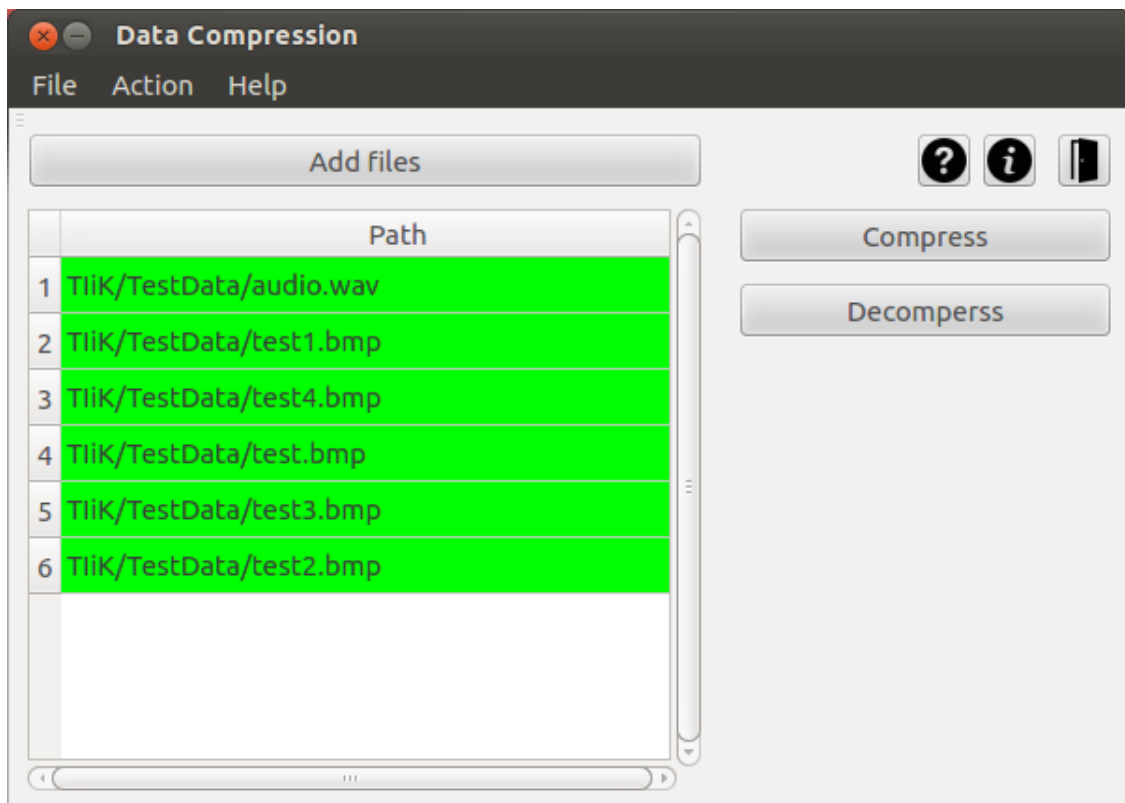
Zrzut ekranu 2: Włączenie informacji o autorach projektu.



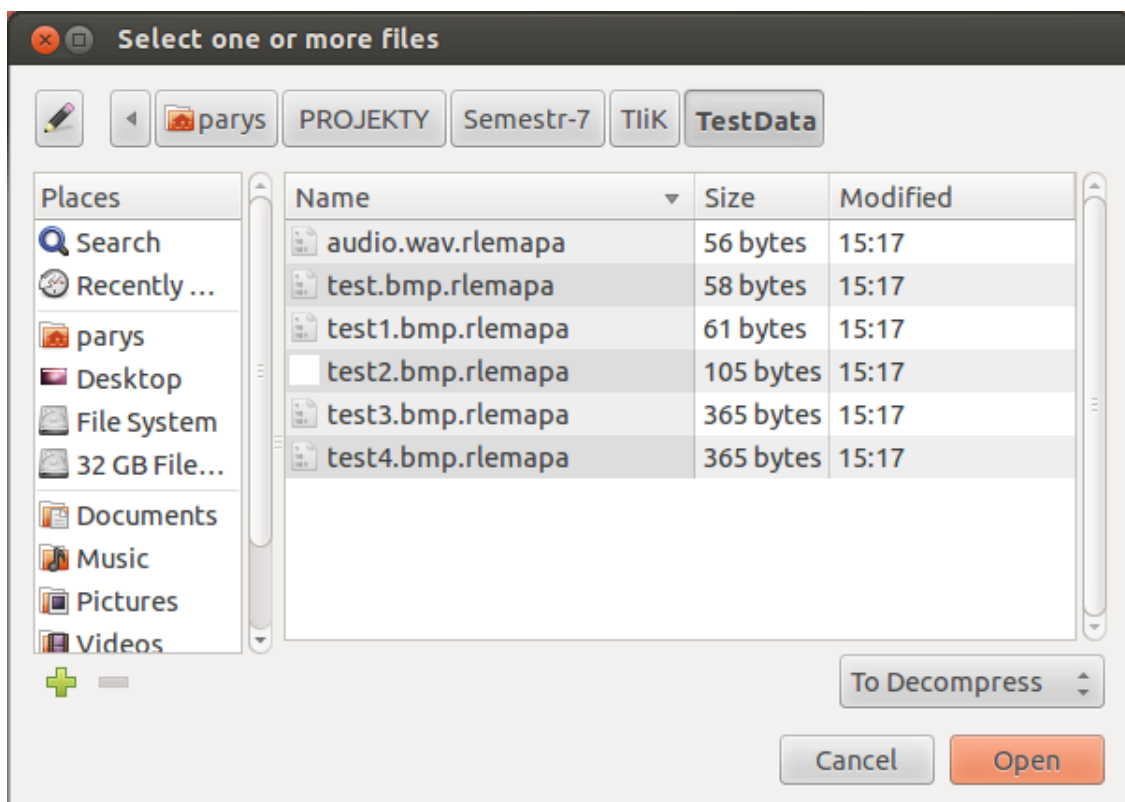
Zrzut ekranu 3: Włączenie okna pomocy.



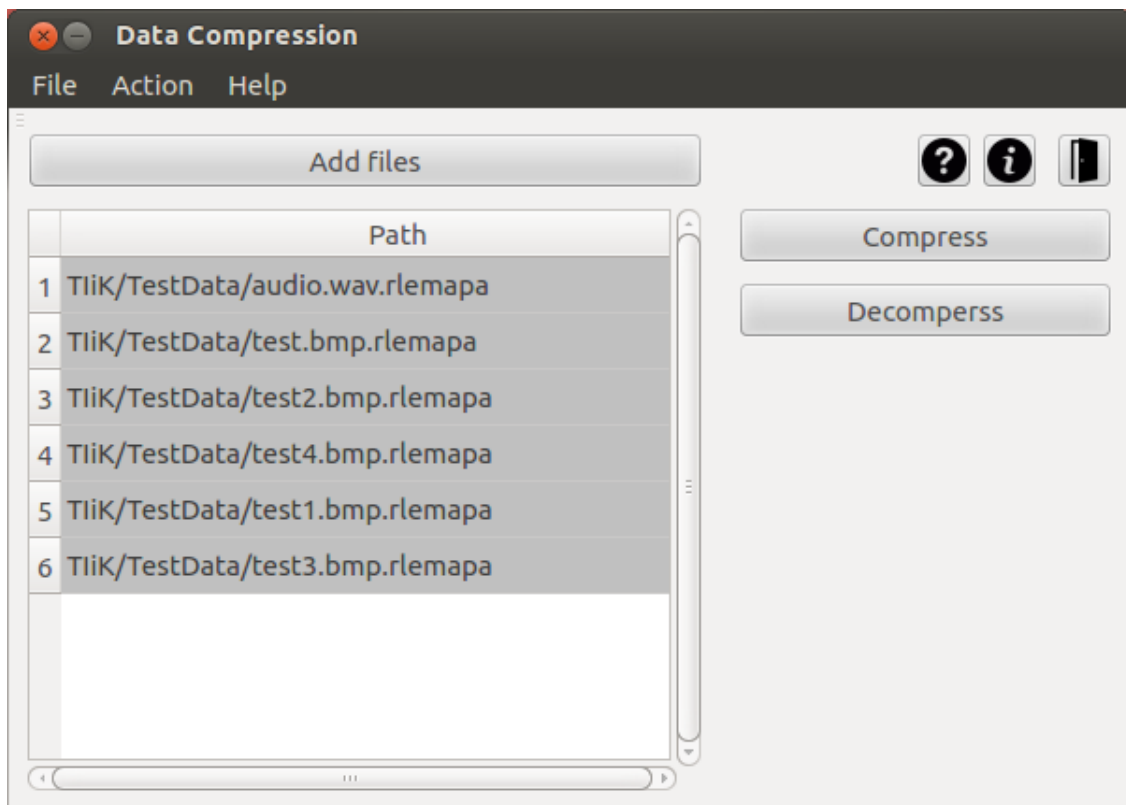
Zrzut ekranu 4: Przetawienie okna dialogowego wyboru plików, które program może skompresować.



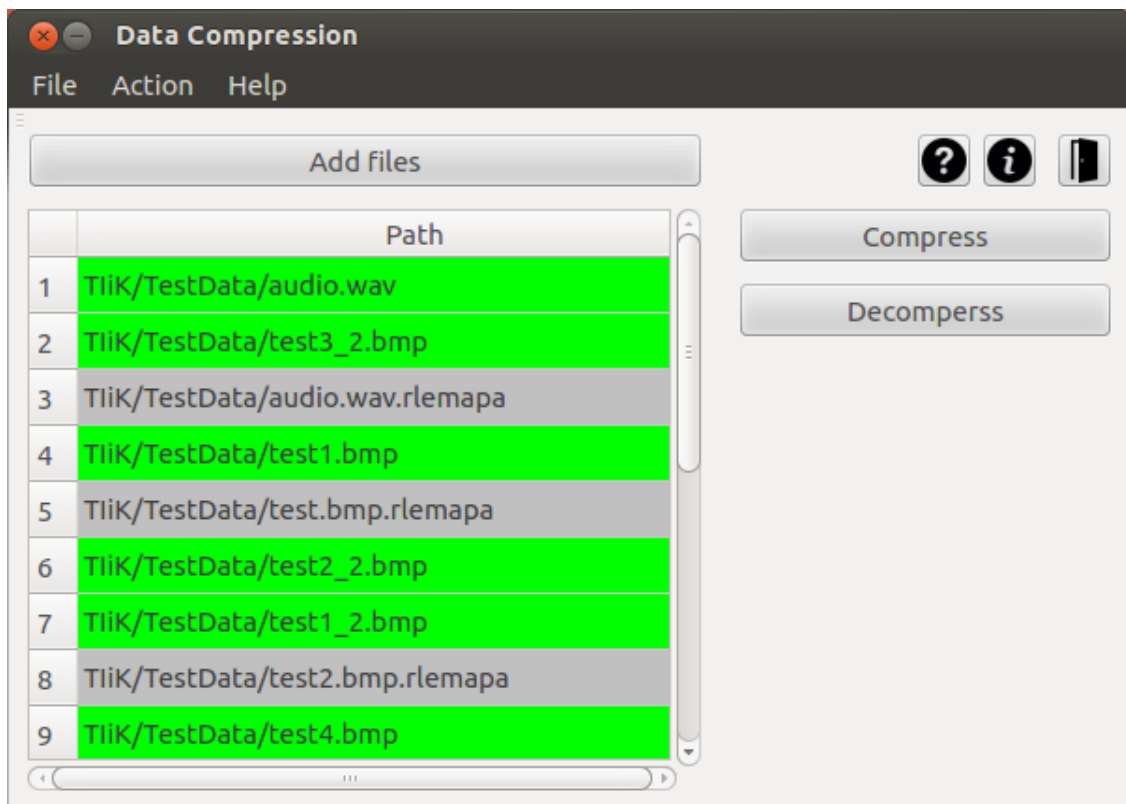
Zrzut ekranu 5: Przedstawienie plików gotowych do kompresji.



Zrzut ekranu 6: Przekształcenie okna dialogowego wyboru plików, które program może zdekompresować.

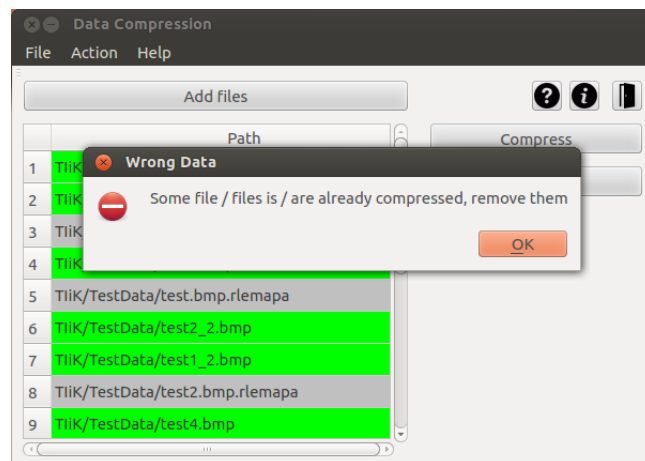
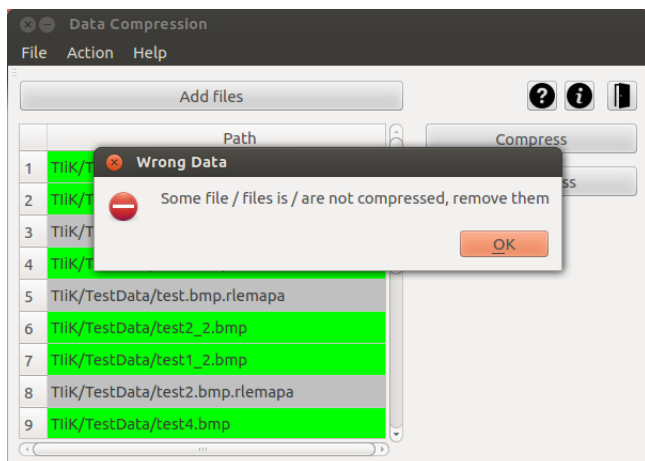


Zrzut ekranu 7: Przedstawienie plików gotowych do dekompresji.



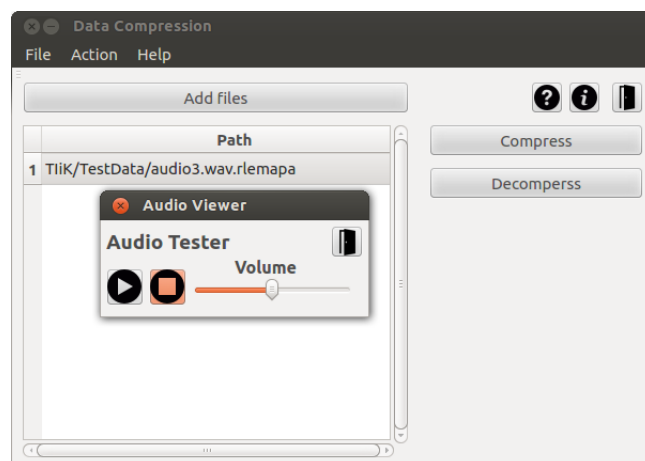
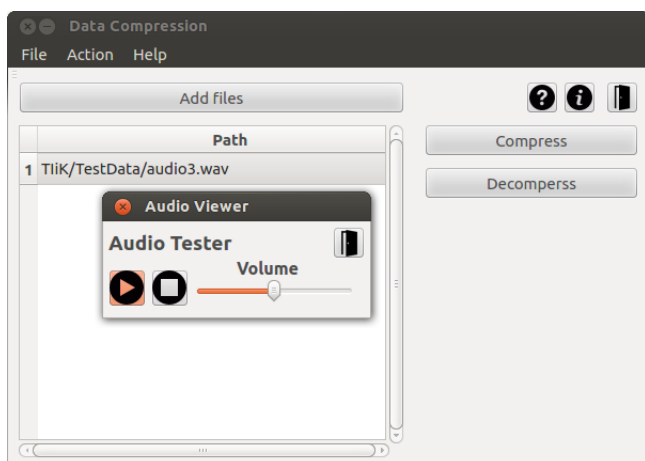
Zrzut ekranu 8: Przedstawienie plików zdekompresowanych i nie skompresowanych

Zrzut ekranu 9: Przedstawienie komunikatów błędów, gdy nakażemy zdekompresować / skompresować pliki „przemieszane”.

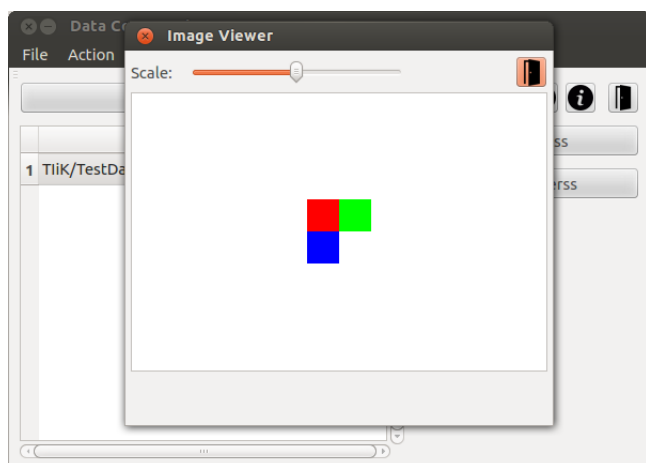


Dodatkowo w programie zaimplementowano mechanizmy „podglądu” przetwarzanych plików.

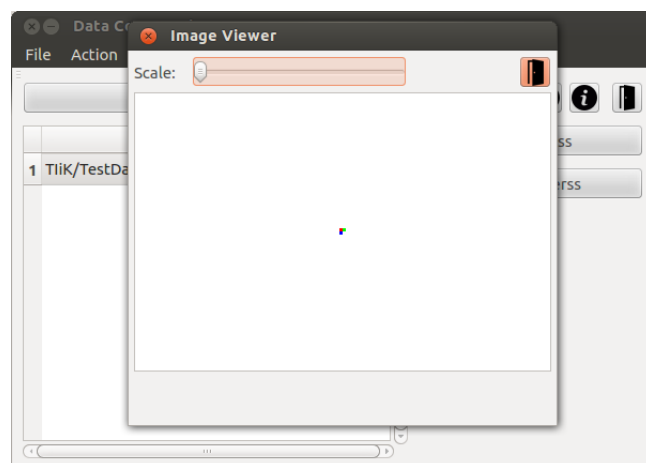
Zrzut ekranu 10: Okno umożliwiające przesłuchanie utworów muzycznych przed i po kompresji.



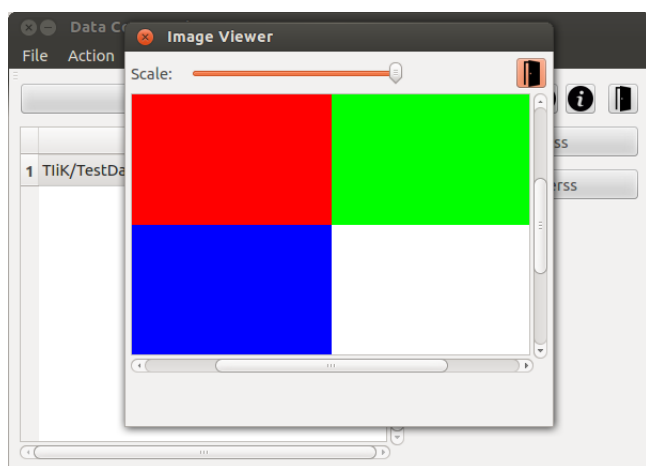
Zrzut ekranu 11: Okno umożliwiające przesłuchanie podgląd plików graficznych.



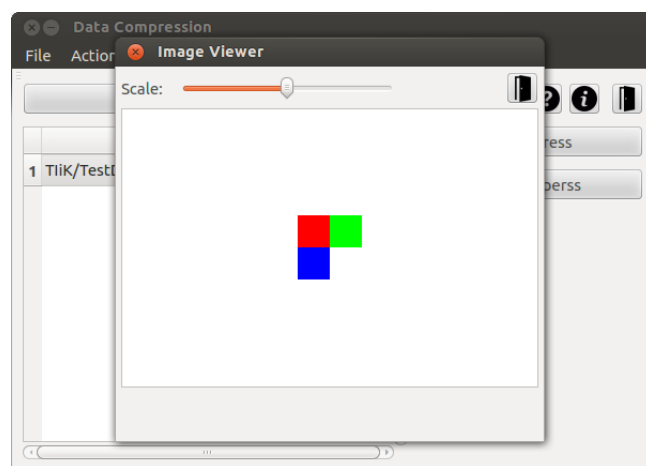
(a) Przed kompresją, bez skali.



(b) Przed kompresją, minimalna skala.



(c) Przed kompresją, maksymalna skala.

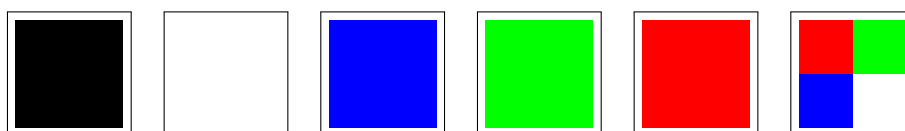


(d) Po kompresji, bez skali.

5 Testy

W celu przeprowadzenia testów wykonano kilka prostych obrazków formatu bmp oraz pobrano z Internetu inne, większe i bardziej skomplikowane obrazki. Dodatkowo do badań pobrano kilka plików dźwiękowych formatu wav z strony <http://download.wavetlan.com/SVV/Media/HTTP/http-wav.htm>[11].

Tabela 2: Przykładowe pliki graficzne (powiększone):



Kod 5: Przedstawienie plików przed kompresją. Kod 6: Przedstawienie plików po kompresji.

1	ROZMIAR	NAZWA	1	ROZMIAR	NAZWA
2	316002	audio2.wav	2	315928	audio2.wav.rlemapa
3	869028	audio3.wav	3	856106	audio3.wav.rlemapa
4	261262	audio4.wav	4	261239	audio4.wav.rlemapa
5	3009870	audio.wav	5	56	audio.wav.rlemapa
6	8533723	highway.wav	6	8447839	highway.wav.rlemapa
7	309464	test10.bmp	7	85103	test10.bmp.rlemapa
8	16000138	test11.bmp	8	8440758	test11.bmp.rlemapa
9	131554	test12.bmp	9	76746	test12.bmp.rlemapa
10	693122	test13.bmp	10	345212	test13.bmp.rlemapa
11	442	test1.bmp	11	61	test1.bmp.rlemapa
12	442	test2.bmp	12	105	test2.bmp.rlemapa
13	442	test3.bmp	13	365	test3.bmp.rlemapa
14	442	test4.bmp	14	365	test4.bmp.rlemapa
15	442	test5.bmp	15	364	test5.bmp.rlemapa
16	170	test6.bmp	16	95	test6.bmp.rlemapa
17	3000122	test7.bmp	17	65	test7.bmp.rlemapa
18	1163198	test8.bmp	18	1163182	test8.bmp.rlemapa
19	44264	test9.bmp	19	5689	test9.bmp.rlemapa
20	126	test.bmp	20	58	test.bmp.rlemapa

Kod 7: Przedstawienie plików przed kompresją. Kod 8: Przedstawienie plików po dekompresji.

1	ROZMIAR	NAZWA	1	ROZMIAR	NAZWA
2	316002	audio2.wav	2	316001	audio2_2.wav
3	869028	audio3.wav	3	869034	audio3_2.wav
4	261262	audio4.wav	4	261262	audio4_2.wav
5	3009870	audio.wav	5	3009873	audio_2.wav
6	8533723	highway.wav	6	8533722	highway_2.wav
7	309464	test10.bmp	7	309464	test10_2.bmp
8	16000138	test11.bmp	8	16000138	test11_2.bmp
9	131554	test12.bmp	9	131599	test12_2.bmp
10	693122	test13.bmp	10	693121	test13_2.bmp
11	442	test1.bmp	11	445	test1_2.bmp
12	442	test2.bmp	12	442	test2_2.bmp
13	442	test3.bmp	13	442	test3_2.bmp
14	442	test4.bmp	14	442	test4_2.bmp
15	442	test5.bmp	15	442	test5_2.bmp
16	170	test6.bmp	16	170	test6_2.bmp
17	3000122	test7.bmp	17	3000125	test7_2.bmp
18	1163198	test8.bmp	18	1163198	test8_2.bmp
19	44264	test9.bmp	19	44270	test9_2.bmp
20	126	test.bmp	20	126	test_2.bmp

5.1 Przedstawienie wyników:

Plik:	audio2.wav	audio3.wav	audio4.wav	audio.wav	highway.wav	test10.bmp
Przed:	316002	869028	261262	3009870	8533723	309464
Po:	315928	856106	261239	56	8447839	85103
Dekom.:	316001	869034	261262	3009873	8533722	309464
%	99	98	99	$1 * 10^{-5}$	98	27

Plik:	test11.bmp	test12.bmp	test13.bmp	test1.bmp	test2.bmp	test3.bmp
Przed:	16000138	131554	693122	442	442	442
Po:	8440758	76746	345212	61	105	365
Dekom.:	16000138	131599	693121	445	442	442
%	52	58	49	13	23	82

Plik:	test4.bmp	test5.bmp	test6.bmp	test7.bmp	test8.bmp	test9.bmp	test.bmp
Przed:	442	442	170	3000122	1163198	44264	126
Po:	365	364	95	65	1163182	5689	58
Dekom.:	442	442	170	3000125	1163198	44270	126
%	82	82	55	$2 * 10^{-5}$	99	12	46

Tabela 3: Porównanie plików graficznych przed i po kompresji (powiększone):

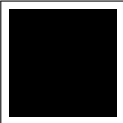

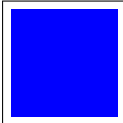
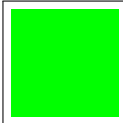
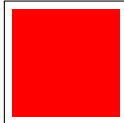
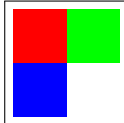


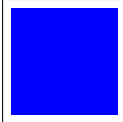
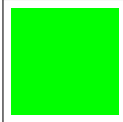
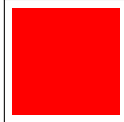
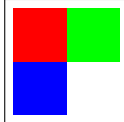
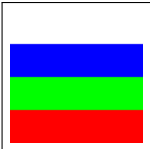
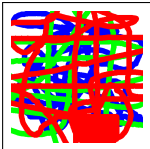



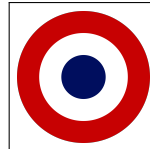



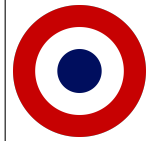


1 PLIK:	test1.bmp	test2.bmp	test3.bmp	test4.bmp	test5.bmp	test6.bmp
PRZED:						
PO:						

Tabela 4: Porównanie plików graficznych przed i po kompresji (pomniejszone):

PLIK:	test8.bmp	test9.bmp	test10.bmp	test11.bmp	test12.bmp	test13.bmp
PRZED						
PO						

Porównania bajtowe wykonano programem vbindiff[12].

Zrzut ekranu 12: Porównanie różnic dwóch tablic bajtowych przed kompresją i po dekompresji.

test1.bmp									
0000	0000:	42	4D	BA	01	00	00	00	00
0000	0010:	00	00	0A	00	00	00	0A	00
0000	0020:	00	00	40	01	00	00	13	0B
0000	0030:	00	00	00	00	00	42	47	52
0000	0040:	00	00	00	00	00	00	00	00
0000	0050:	00	00	00	00	00	00	00	00
0000	0060:	00	00	00	00	00	00	00	00
0000	0070:	00	00	00	00	00	00	00	00
0000	0080:	00	00	00	00	00	00	00	00
0000	0090:	00	00	00	00	00	00	00	00
0000	00A0:	00	00	00	00	00	00	00	00
0000	00B0:	00	00	00	00	00	00	00	00
0000	00C0:	00	00	00	00	00	00	00	00
0000	00D0:	00	00	00	00	00	00	00	00
0000	00E0:	00	00	00	00	00	00	00	00
0000	00F0:	00	00	00	00	00	00	00	00
test1_2.bmp									
0000	0000:	42	4D	BA	01	00	00	00	00
0000	0010:	00	00	0A	00	00	00	0A	00
0000	0020:	00	00	40	01	00	00	13	0B
0000	0030:	00	00	00	00	00	42	47	52
0000	0040:	00	00	00	00	00	00	00	00
0000	0050:	00	00	00	00	00	00	00	00
0000	0060:	00	00	00	00	00	00	00	00
0000	0070:	00	00	00	00	00	00	00	00
0000	0080:	00	00	00	00	00	00	00	00
0000	0090:	00	00	00	00	00	00	00	00
0000	00A0:	00	00	00	00	00	00	00	00
0000	00B0:	00	00	00	00	00	00	00	00
0000	00C0:	00	00	00	00	00	00	00	00
0000	00D0:	00	00	00	00	00	00	00	00
0000	00E0:	00	00	00	00	00	00	00	00
0000	00F0:	00	00	00	00	00	00	00	00
Arrow keys move F find RET next difference ESC quit T move top									
C ASCII/EBCDIC E edit file G goto position Q quit B move bottom									

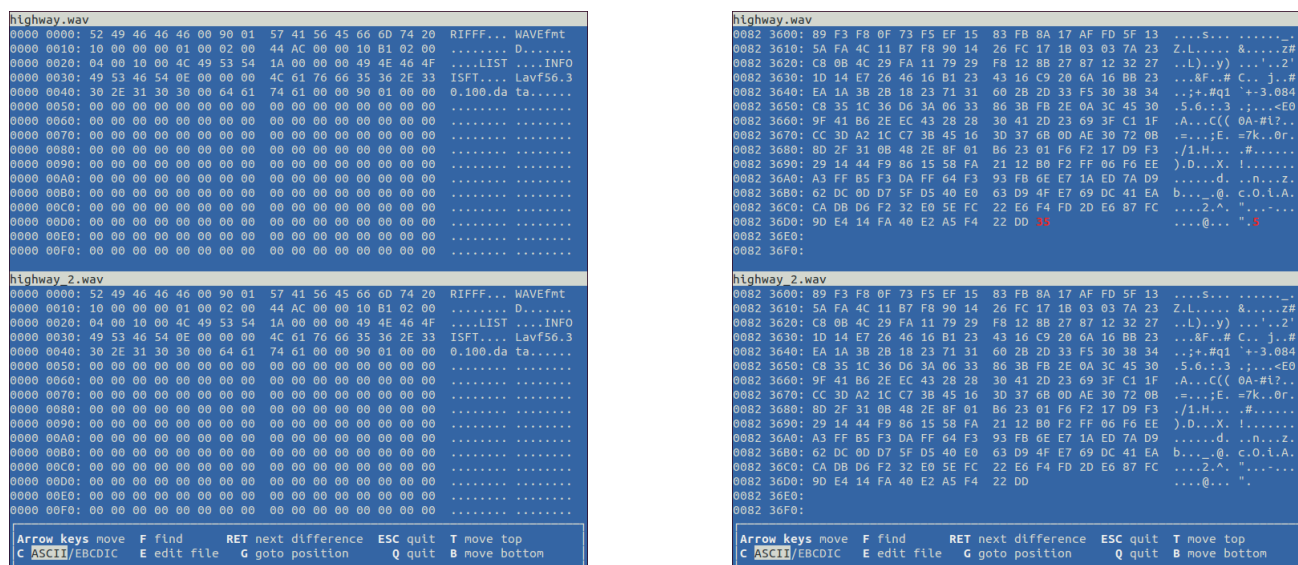
test1.bmp									
0000	0100:	00	00	00	00	00	00	00	00
0000	0110:	00	00	00	00	00	00	00	00
0000	0120:	00	00	00	00	00	00	00	00
0000	0130:	00	00	00	00	00	00	00	00
0000	0140:	00	00	00	00	00	00	00	00
0000	0150:	00	00	00	00	00	00	00	00
0000	0160:	00	00	00	00	00	00	00	00
0000	0170:	00	00	00	00	00	00	00	00
0000	0180:	00	00	00	00	00	00	00	00
0000	0190:	00	00	00	00	00	00	00	00
0000	01A0:	00	00	00	00	00	00	00	00
0000	01B0:	00	00	00	00	00	00	00	00
0000	01C0:								
0000	01D0:								
0000	01E0:								
0000	01F0:								
test1_2.bmp									
0000	0100:	00	00	00	00	00	00	00	00
0000	0110:	00	00	00	00	00	00	00	00
0000	0120:	00	00	00	00	00	00	00	00
0000	0130:	00	00	00	00	00	00	00	00
0000	0140:	00	00	00	00	00	00	00	00
0000	0150:	00	00	00	00	00	00	00	00
0000	0160:	00	00	00	00	00	00	00	00
0000	0170:	00	00	00	00	00	00	00	00
0000	0180:	00	00	00	00	00	00	00	00
0000	0190:	00	00	00	00	00	00	00	00
0000	01A0:	00	00	00	00	00	00	00	00
0000	01B0:	00	00	00	00	00	00	00	00
0000	01C0:						00	00	00
0000	01D0:								
0000	01E0:								
0000	01F0:								
Arrow keys move F find RET next difference ESC quit T move top									
C ASCII/EBCDIC E edit file G goto position Q quit B move bottom									

Zrzut ekranu 13: Porównanie różnic dwóch tablic bajtowych przed kompresją i po dekompresji.

audio2.wav									
0000	0000:	52	49	46	46	5A	D2	04	00
0000	0010:	10	00	00	00	01	00	02	00
0000	0020:	04	00	10	00	64	61	74	61
0000	0030:	35	00	31	00	39	00	78	00
0000	0040:	52	00	A8	00	61	00	B5	00
0000	0050:	92	00	48	01	87	00	6F	01
0000	0060:	B7	00	3D	02	DA	00	57	02
0000	0070:	2A	01	7E	03	33	01	C7	03
0000	0080:	93	01	E6	04	D5	01	71	05
0000	0090:	7B	02	09	07	BE	02	C3	07
0000	00A0:	5B	03	F2	08	E2	03	29	0A
0000	00B0:	08	04	55	0C	7D	04	2E	0B
0000	00C0:	AB	FC	F4	F2	32	FC	B6	F0
0000	00D0:	2E	FC	42	EF	DC	FB	C7	ED
0000	00E0:	BE	FB	87	EA	8B	FB	9D	E9
0000	00F0:	DF	FB	7B	E6	E8	FB	E8	E5
audio2_2.wav									
0000	0000:	52	49	46	46	5A	D2	04	00
0000	0010:	10	00	00	00	01	00	02	00
0000	0020:	04	00	10	00	64	61	74	61
0000	0030:	35	00	31	00	39	00	78	00
0000	0040:	52	00	A8	00	61	00	B5	00
0000	0050:	92	00	48	01	87	00	6F	01
0000	0060:	B7	00	3D	02	DA	00	57	02
0000	0070:	2A	01	7E	03	33	01	C7	03
0000	0080:	93	01	E6	04	D5	01	71	05
0000	0090:	7B	02	09	07	BE	02	C3	07
0000	00A0:	5B	03	F2	08	E2	03	29	0A
0000	00B0:	08	04	55	0C	7D	04	2E	0B
0000	00C0:	AB	FC	F4	F2	32	FC	B6	F0
0000	00D0:	2E	FC	42	EF	DC	FB	C7	ED
0000	00E0:	BE	FB	87	EA	8B	FB	9D	E9
0000	00F0:	DF	FB	7B	E6	E8	FB	E8	E5
Arrow keys move F find RET next difference ESC quit T move top									
C ASCII/EBCDIC E edit file G goto position Q quit B move bottom									

audio2.wav									
0004	D200:	00	40	00	00	00	5C	00	00
0004	D210:	6B	2C	54	50	45	31	00	00
0004	D220:	65	77	61	76	65	73	61	6D
0004	D230:	54	44	52	43	00	00	00	05
0004	D240:	49	54	32	00	00	00	19	00
0004	D250:	20	54	47	31	30	30	20	53
0004	D260:	43	00						
0004	D270:								
0004	D280:								
0004	D290:								
0004	D2A0:								
0004	D2B0:								
0004	D2C0:								
0004	D2D0:								
0004	D2E0:								
0004	D2F0:								
audio2_2.wav									
0004	D200:	00	40	00	00	00	5C	00	00
0004	D210:	6B	2C	54	50	45	31	00	00
0004	D220:	65	77	61	76	65	73	61	6D
0004	D230:	54	44	52	43	00	00	00	05
0004	D240:	49	54	32	00	00	00	19	00
0004	D250:	20	54	47	31	30	30	20	53
0004	D260:	43							
0004	D270:								
0004	D280:								
0004	D290:								
0004	D2A0:								
0004	D2B0:								
0004	D2C0:								
0004	D2D0:								
0004	D2E0:								
0004	D2F0:								
Arrow keys move F find RET next difference ESC quit T move top									
C ASCII/EBCDIC E edit file G goto position Q quit B move bottom									

Zrzut ekranu 14: Porównanie różnic dwóch tablic bajtowych przed kompresją i po dekompresji.



6 Wnioski

Algorytm **Run-Length Encoding** doskonale się sprawdza do kompresji plików graficznych monochromatycznych (jedno kolorowych) - kompresuje pliki na poziomie 20 – 50%. Przy bardziej złożonych - bardziej kolorowych - plikach graficznych kompresja spada do poziomu 80 – 99%. Pliki dźwiękowe, z racji swojej zawłości również nie są podatne na kompresję tym algorytmem.

Algorytm jest prosty w implementacji i obsłudze oraz dla niektórych plików może dać „zachwycające” rezultaty - kompresja na poziomie tysięcznych części pliku wejściowego. Jednakże nie gwarantuje on „oszczędności” miejsca - kompresji.

W procesie kompresji / dekompresji nie powstały żadne znaczące błędy. „Przeinaczenia” bajtowe powstałe wskutek „wypełnienia” bajtami z dekompresowanych plików nie wpływają na obiekt końcowy.

Niewielkie uszkodzenia plików powstałe podczas testowania wystąpiły na skutek przeciążenia systemu innymi zadaniami.

Literatura

- [1] Kurs L^AT_EX w π^e minut http://www.fuw.edu.pl/~kostecki/kurs_latexa.pdf.
- [2] Program Texmaker 4.5 <http://www.xmlmath.net/texmaker/>.
- [3] ShareLaTeX online LaTeX editor <https://www.sharelatex.com/>.
- [4] Qt Creator – wieloplatformowe środowisko programistyczne <http://www.qt.io/>.
- [5] GCC, the GNU Compiler Collection <https://gcc.gnu.org/>.
- [6] Git rozproszony system kontroli wersji <http://git-scm.com/>.
- [7] GitHub Web-based Git repository <https://github.com/>.
- [8] Run-length encodings - S. W. Golomb (1966); IEEE Trans Info Theory 12(3):399 http://urchin.earth.li/~twic/Golombs_Original_Paper/.
- [9] Variable-length codes for data compression / David Salomon, London : Springer, 2007.
- [10] The largest database of free vector icons - flaticon <http://www.flaticon.com/>.

- [11] Sample WAV files <http://download.wavetlan.com/SVV/Media/HTTP/http-wav.htm>.
- [12] vbindiff - hexadecimal file display and comparison <http://manpages.ubuntu.com/manpages/hardy/man1/vbindiff.1.html>.