LZW (Lempel-Ziv-Welch) – metoda strumieniowej bezstratnej kompresji słownikowej. Najprościej rzecz ujmując algorytm ten buduje pewnego rodzaju słownik wartości, a następnie koduje dane wejściowe za pomocą indeksów elementów tegoż słownika.

Przykład 1:

wabbawabba <- wiadomość wejściowa

Słownik dla podanych danych:

Indeks	Wartość
1	a
2	b
3	w
4	wa
5	ab
6	bb
7	ba
8	aw
9	wab
10	bba

3 1 2 2 1 4 6 1 <- zakodowana wiadomość

Algorytm:

1. Budujemy słownik podstawowy – każdy znak padający w wiadomości umieszczamy w słowniku najlepiej w kolejności alfabetycznej. W powyższym przykładzie padają tylko trzy litery.

Indeks	Wartość
1	A
2	B
3	W

2. Teraz przystępujemy do właściwego algorytmu. Do zmiennej **c** przypisujemy pierwszy znak wiadomości

   c = message[0]; //c = w

3. Dla **i** od **1** do **n-1** gdzie **n** to długość wiadomości: do zmiennej **s** przypisujemy kolejny znak wiadomości

   s = message[i]; //s = a

4. Jeżeli ciąg **c + s** znajduje się w słowniku do **c** przypisujemy **c + s**
   c = c + s;

5. Jeżeli **c + s** nie ma w słowniku dodajemy **c + s** do słownika, zapisujemy indeks **c**. Następnie do **c** przypisujemy **s**.

   c = s;

6. Jeżeli **i < n** wracamy do punktu 3. W przeciwnym razie zapamiętujemy indeks aktualnej wartości **c** w słowniku.

7. Zapamiętane wartości to nasz kod. Pobranie ze słownika wartości pod zapamiętanymi indeksami a następne ich sklejenie odkoduje wiadomość początkową.

Przykład 2:

abccd_abccd_acd_acd_acd_ <- wiadomość do zakodowania

Przejście krok po kroku:

c	s	c + s	zapamiętany indeks	wartość dodana do słownika	komentarz
				1. a 2. b 3. c 4. d 5. _	inicjacja podstawowego słownika
a	b	ab	1 – indeks ‘a’	6. ab	‘ab’ do słownika, c = ‘b’
b	c	bc	2 – indeks ‘b’	7. bc	‘bc’ do słownika, c = ‘c’
c	c	cc	3 – indeks ‘c’	8. cc	‘cc’ do słownika, c = ‘c’
c	d	cd	3 – indeks ‘c’	9. cd	‘cd’ do słownika, c =’d’
d	_	d_	4 - indeks ‘d’	10. d_	‘d_’ do słownika, c =’_’
_	a	_a	5 – indeks ‘_’	11. _a	‘_a’ do słownika, c=’a’
a	b	ab			‘ab’ w słowniku, c=’ab’
ab	c	abc	6 – indeks ‘ab’	12. abc	‘abc’ do słownika, c = ‘c’
c	c	cc			‘cc’ w słowniku, c=’cc’
cc	d	ccd	8 – indeks ‘cc’	13. ccd	‘ccd’’ do słownika, c=’d’
d	_	d_			‘d_’ w słowniku, c=’d_’
d_	a	d_a	10 – indeks ‘d_’	14. d_a	‘d_a’ do słownika, c=’a’
a	c	ac	1 – indeks ‘a’	15. ac	‘ac’ do słownika, c=’c’
c	d	cd			‘cd’ w słowniku, c=’cd’
cd	_	cd_	9 – indeks ‘cd’	16. cd_	‘cd_’ do słownika, c=’_’
_	a	_a			‘_a’ w słowniku, c=’_a’
_a	c	_ac	11 – indeks ‘_a’	17. _ac	‘_ac’ do słownika, c=’c’
c	d	cd			‘cd’ w słowniku, c=’cd’

cd	_	cd_			‘cd_’ w słowniku, c=’cd_’
cd_	a	cd_a	16 – indeks ‘cd_’	18. cd_a	‘cd_a’ do słownika, c=’a’
a	c	ac			‘ac’ w słowniku, c = ‘ac’
ac	d	acd	15 – indeks ‘ac’	19. acd	‘acd’ do słownika, c = ‘d’
d	_	d_	10 – indeks ‘d_’		koniec

Czytając kolumnę czwartą z góry w dół otrzymujemy kod: 1, 2,3,3,4,5,6,7,10,1,9,11,16,15,10

Kolumna piąta to wygenerowany słownik.

Zadanie:

Zaimplementuj algorytm LZW:

Na ocenę dst:

- program pobiera wiadomość

- program generuje słownik

- program wypisuje zakodowaną wiadomość

Na ocenę db:

- program pobiera wiadomość

- program generuje słownik i zapisuje go w pliku tekstowym (linia po linii, indeks – wartość)

- program wypisuje zakodowaną wiadomość

Na ocenę bdb:

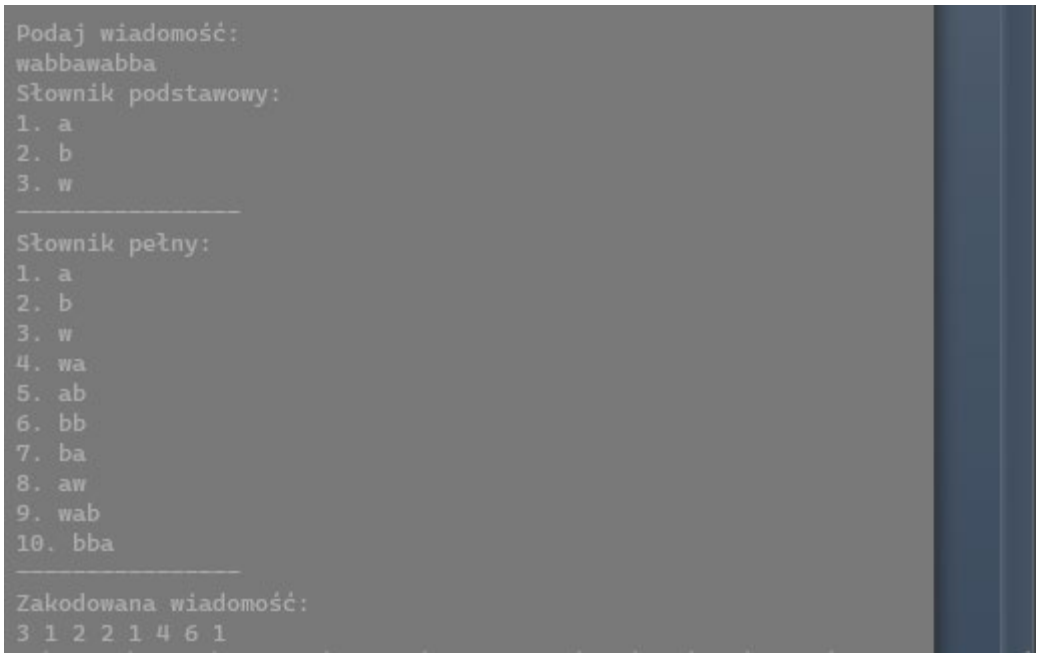
- program pobiera wiadomość

- program generuje słownik i zapisuje go w pliku tekstowym (linia po linii, indeks – wartość)

- program wypisuje zakodowaną wiadomość

* napisz także program, który na bazie wygenerowanego wcześniej pliku tekstowego ze słownikiem dekoduje wiadomość

Przykładowy output programu na ocenę dst:


Podaj wiadomość:
wabbawabba
Słownik podstawowy:
1. a
2. b
3. w
Słownik pełny:
1. a
2. b
3. w
4. wa
5. ab
6. bb
7. ba
8. aw
9. wab
10. bba
Zakodowana wiadomość:
3 1 2 2 1 4 6 1