Akademia Nauk Stosowanych - Teoretyczne i technologiczne podstawy multimediów - laboratorium			
Temat: LZW – metoda strumieniowej bezstratnej kompresji słownikowej.			Symbol: TiTPM
Nazwisko i imię: Fyda Kamil		Ocena sprawozdania	Zaliczenie:
Data wykonania ćwiczenia: 15.11.2022r.	Grupa: L1		

1. Opis teoretyczny:

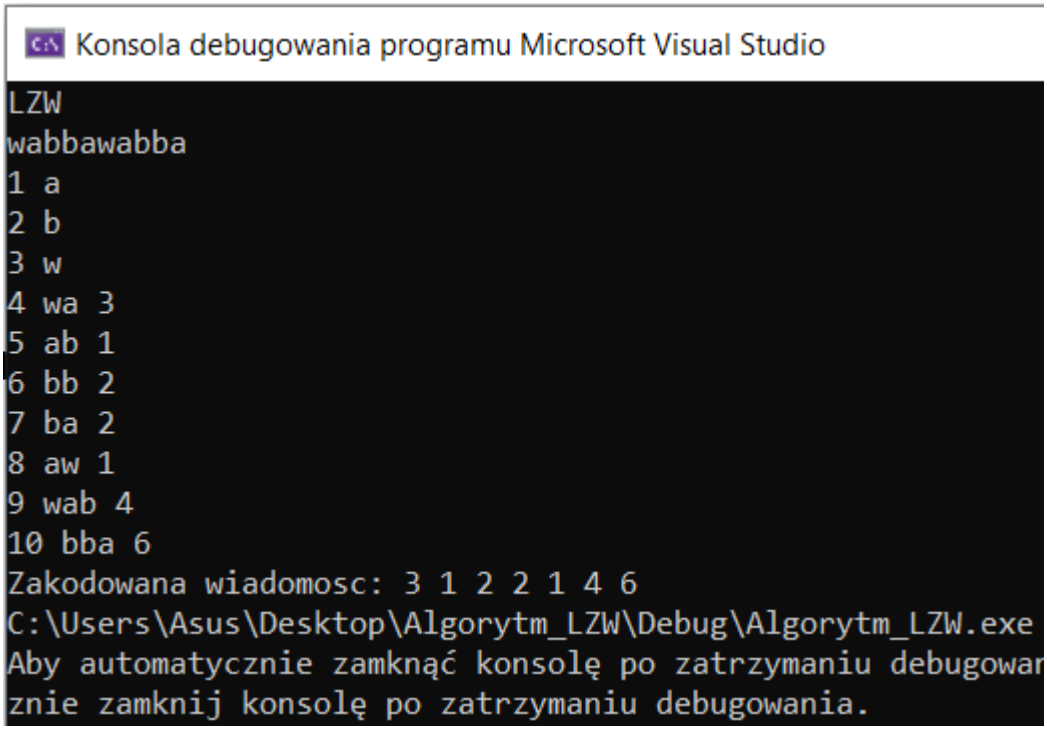
LZW (Lempel-Ziv-Welch) – metoda strumieniowej bezstratnej kompresji słownikowej. Najprościej rzecz ujmując algorytm ten buduje pewnego rodzaju słownik wartości, a następnie koduje dane wejściowe za pomocą indeksów elementów tegoż słownika. Algorytm LZW jest dosyć prosty. Wykorzystuje ona fakt, że na obrazie istnieją piksele o powtarzających się kolorach i tworzy na tej podstawie słownik takich powtarzających się ciągów znaków. Całe połączenie zastępowane jest skrótem. Takie przypisanie jest oczywiście zapisywane i dzięki temu dekompresor może odtworzyć plik w postaci nie zmienionej. Bardzo dobrze sprawdza się przy obrazach zawierających duże obszary jednolitego koloru lub tam gdzie istnieje wiele powtarzających się schematów/elementów na obrazie (8-bitowe). Podczas dekompresji czyli otwierania pliku dzieje się sytuacja odwrotna – za pomocą słownika skróty są podmieniane na powtarzające się ciągi znaków. Dzięki temu możemy odtworzyć plik w postaci niezmienionej. Identyczny jak przed kompresją. Jedyny minus jest taki, że słabo radzi sobie z plikami 16-bitowymi. Dzięki LZW można uzyskać plik mniejszy nawet o połowę, jest to więc bardzo zadowalająca metoda kompresji.

2. Kod programu:

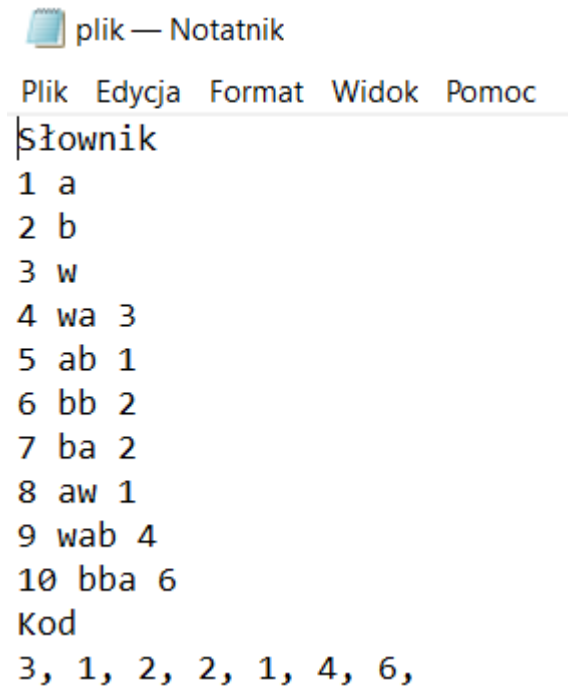
```cpp
#include <iostream>
#include <string>
#include <fstream>
using namespace std;
int main()
{
    cout << "LZW\n";
    string tekst = "wabbawabba";
    cout << tekst << endl;
    //getline(cin, tekst);
    string tab[30][3];
    int nast = 0;
    for (int i = 0; i < 30; i++)
    {
        tab[i][0] = "";
        tab[i][1] = "";
        tab[i][2] = "";
    }
    //WPISANIE DO TABLICY WSZYSTKICH WYSTĘPUJĄCYCH W TEKSCIE LITER
    for (int i = 0; i < tekst.length(); i++)
    {
        int licznik = 0;
        string litera = "";
        char r = tekst.at(i);
        litera += r;
        for (int j = 0; j < 30; j++)
        {
            if (litera == tab[j][1]) licznik++;
        }
        if (licznik == 0)
        {
            tab[i][1] = litera;
            tab[i][0] = to_string(i + 1);
            nast++;
        }
    }
    //Posortowanie tablicy
    for (int i = 0; i < 30; i++)
    {
        for (int j = i + 1; j < 30; j++)
        {
            if (tab[j][1] < tab[i][1] && tab[j][1] != "")
            {
                string temp1 = tab[i][1];
                tab[i][1] = tab[j][1];
                tab[j][1] = temp1;
            }
        }
    }
    string c = "";
    string zapamietana = "";
    char litera = tekst.at(0);
    c += litera; //wartosc c - pierwsza litera wiadomosci
    for (int i = 1; i < tekst.length(); i++)
    {
        for (int j = 0; j < 30; j++)
        {
            //Zapamiętanie indeksu aktualnej wartosci c w słowniku
            if (c == tab[j][1]) zapamietana = tab[j][0];
        }
        int licznik = 0;
        string s = "";
        char litera2 = tekst.at(i);
        s = litera2;
        for (int j = 0; j < 30; j++)
        {
            if ((c + s) == tab[j][1]) licznik++;
        }
        if (licznik > 0) c = c + s;
        else
        {
            tab[nast][2] = zapamietana;
```

```
            tab[nast][1] = c + s;
            tab[nast][0] = to_string(nast + 1);
            nast++;
            c = s;
        }
    }
    //Wyświetlenie gotowego słownika
    for (int i = 0; i < 30; i++)
    {
        if (tab[i][1] != "")
        {
            cout << tab[i][0] << " " << tab[i][1] << " " << tab[i][2] << endl;
        }
    }
    //Wypisanie zakodowanej wiadomości
    cout << "Zakodowana wiadomosc: ";
    for (int i = 0; i < 30; i++)
    {
        if (tab[i][2] != "")
        {
            cout << tab[i][2] << " ";
        }
    }
    //Zapisanie wiadomości do pliku
    fstream plik("plik.txt", ios::out);
    if (plik.good())
    {
        plik << "Słownik" << endl;
        plik.flush();
        for (int i = 0; i < 30; i++)
        {
            if (tab[i][1] != "")
            {
                plik << tab[i][0] << " " << tab[i][1] << " " << tab[i][2] << endl;
                plik.flush();
            }
        }
        plik << "Kod" << endl;
        plik.flush();
        for (int i = 0; i < 30; i++)
        {
            if (tab[i][2] != "")
            {
                plik << tab[i][2] << ", ";
                plik.flush();
            }
        }
        plik.close();
    }
}
```

3. **Wynik działania programu:**



4. **Wynik działania zapisywania do pliku:**



Wnioski:

Program wyświetla odpowiednie przyporządkowania słownikowe zgodne z algorytmem LZW oraz zapisuje otrzymane dane do pliku „plik.txt” dlatego można uznać program za działający w pełni poprawnie w porównaniu do wcześniejszych założeń.