Teoretyczne i technologiczne podstawy multimediów	**TITPM05**	**Akademia Nauk Stosowanych**
13.11.2022	**L2**	**Szymon Zwoliński**

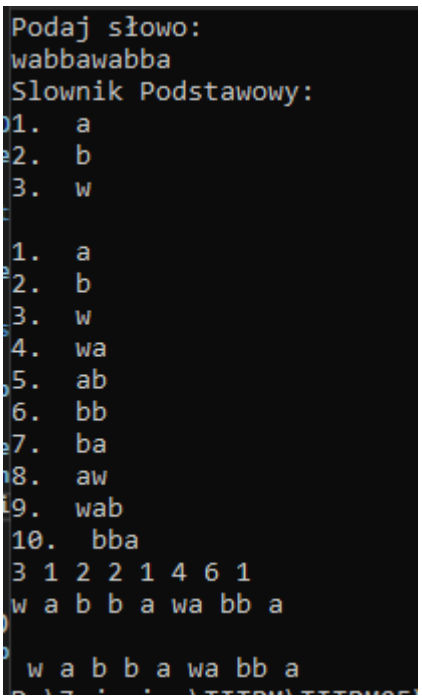
1. Lempel-Ziv-Welch:

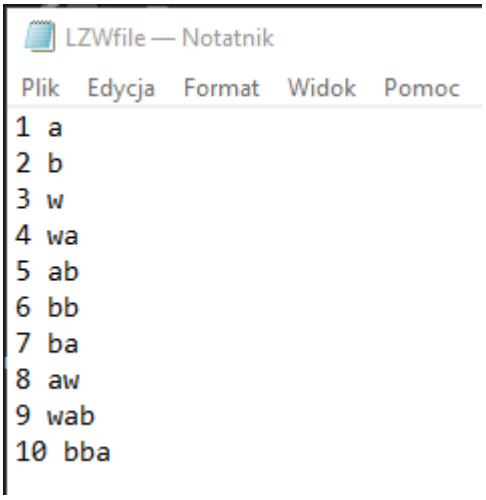
Algorytm Lempel-ZIv-Welcha zakłada utworzenie słownika, który umożliwia bezstratną kompresję danych.

W pojedynczym kroku algorytmu wyszukiwany jest w słowniku najdłuższy prefiks niezakodowanych jeszcze danych. Na wyjście wypisywany jest wówczas kod związany z tym słowem, zaś do słownika dodawana nowa pozycja: konkatenacja słowa i pierwszej niedopasowanej litery.

Algorytm przebiega następująco:

Wypełnij słownik alfabetem źródła informacji.
c := pierwszy symbol wejściowy
Dopóki są dane na wejściu:
Wczytaj znak s.
Jeżeli ciąg c + s znajduje się w słowniku, przedłuż ciąg c, tj. c := c + s
Jeśli ciągu c + s nie ma w słowniku, wówczas:
wypisz kod dla c (c znajduje się w słowniku)
dodaj ciąg c + s do słownika
przypisz c := s.
Na końcu wypisz na wyjście kod związany c.


Podaj słowo:
wabbawabba
Slownik Podstawowy:
1. a
2. b
3. w
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
w a b b a wa bb a
w a b b a wa bb a


LZWfile — Notatnik
Plik Edycja Format Widok Pomoc
1 a
2 b
3 w
4 wa
5 ab
6 bb
7 ba
8 aw
9 wab
10 bba

```
public static void LZW(string input)
{
    string c = input[0].ToString();
    int lenOfInput = input.Count();
    int lenOfDic = dictionary.Last().index;

    string s = string.Empty;
    for (int i = 1; i < lenOfInput; i++)
    {
        s = input[i].ToString();
        if (dictionary.Any(x => x.value == (c + s)))
        {
            c = c + s;
        }
        else
        {
            code.Add(dictionary[dictionary.FindIndex(x => x.value == c)].index);
            lenOfDic++;
            dictionary.Add(new Dic { index = lenOfDic, value = (c + s) });
            c = s;
        }

    }
    code.Add(dictionary[dictionary.FindIndex(x => x.value == c)].index);
}
```

```
#region ProgramToDecode
public static List<Dic> listToRecreateDic= new List<Dic>();
1 odwołanie
public static void FromFile()
{
    int len;
    int indexOfSpace;
    int counterOfLines=0;
    int count = 1;
    foreach(string line in File.ReadLines("LZWfile.txt"))
    {
        counterOfLines++;
    }
    foreach (string line in File.ReadLines("LZWfile.txt"))
    {
        if (count != counterOfLines)
        {
            len = line.Length - 1;
            indexOfSpace = line.IndexOf(' ');
            listToRecreateDic.Add(new Dic
            {
                index = int.Parse(line.Substring(0, indexOfSpace)),
                value = line.Substring(indexOfSpace, len)
            });
            count++;
        }
        else
        {
            break;
        }


    }
}
1 odwołanie
public static void DecodeFromList(List<int> code)
{
    Console.Write("\n");
    Console.Write("\n");
    foreach (var codedChar in code)
    {
        Console.Write(listToRecreateDic.FirstOrDefault(x => x.index == codedChar).value);
    }
}
#endregion
```